

Advant[®] OCS
with Master software

User Defined PC Elements
Advant[®] Controller 400 Series

User's Guide

3BSE 009 739R0001 Rev A



Use of **DANGER**, **WARNING**, **CAUTION**, and **NOTE**

This publication includes, **DANGER**, **WARNING**, **CAUTION**, and **NOTE** information where appropriate to point out safety related or other important information.

DANGER	Hazards which could result in severe personal injury or death
WARNING	Hazards which could result in personal injury
CAUTION	Hazards which could result in equipment or property damage
NOTE	Alerts user to pertinent facts and conditions

Although **DANGER** and **WARNING** hazards are related to personal injury, and **CAUTION** hazards are associated with equipment or property damage, it should be understood that operation of damaged equipment could, under certain operational conditions, result in degraded process performance leading to personal injury or death. Therefore, comply fully with all **DANGER**, **WARNING**, and **CAUTION** notices.

TRADEMARKS

Advant and AdvaBuild are registered trademarks of ABB Asea Brown Boveri Ltd. Switzerland

Advant Controller is a trademark of ABB Asea Brown Boveri Ltd. Switzerland

ABB Master, ABB MasterPiece and ABB MasterAid are registered trademarks of ABB Asea Brown Boveri Ltd. Switzerland

NOTICE

The information in this document is subject to change without notice and should not be construed as a commitment by ABB Industrial Systems AB. ABB Industrial Systems AB assumes no responsibility for any errors that may appear in this document.

In no event shall ABB Industrial Systems AB be liable for direct, indirect, special, incidental or consequential damages of any nature or kind arising from the use of this document, nor shall ABB Industrial Systems AB be liable for incidental or consequential damages arising from use of any software or hardware described in this document.

This document and parts thereof must not be reproduced or copied without ABB Industrial Systems AB's written permission, and the contents thereof must not be imparted to a third party nor be used for any unauthorized purpose.

The software described in this document is furnished under a license and may be used, copied, or disclosed only in accordance with the terms of such license.

Copyright © ABB Industrial Systems AB 1998.

TABLE OF CONTENTS

Chapter 1 - Introduction

1.1	Before You Begin	1-1
1.2	Equipment Requirements.....	1-1
1.2.1	Hardware.....	1-1
1.2.2	Software	1-1
1.2.3	Tools.....	1-2
1.3	Conventions	1-2
1.4	Related Documents	1-3
1.5	Terminology.....	1-3
1.6	Release History	1-5

Chapter 2 - Installation

2.1	Introduction.....	2-1
2.2	Control Solution.....	2-1
2.3	Creation of User Elements in FCB	2-2
2.3.1	Definitions	2-2
2.3.2	User and Interface Elements in FCB	2-4
2.4	Creation of Library Program and Source Code	2-5
2.5	User Elements in the Controller.....	2-6
2.5.1	Definitions	2-6
2.5.2	Installation in the Controller	2-7
2.6	Applying User Elements.....	2-8

Chapter 3 - Configuration/Application Building

3.1	User Element Description	3-1
3.2	Terminal Descriptors.....	3-4
3.3	Interface Element Description	3-5
3.4	Target Description (TED)	3-7
3.4.1	Definition	3-7
3.4.2	Creating a TED File.....	3-8
3.5	Installing PED Libraries in FCB.....	3-10
3.6	Create Library Program	3-12

Chapter 4 - User Elements in the Controller

4.1	Introduction.....	4-1
4.2	DB Element AAB_FILE.....	4-1
4.3	Applying FPS_WORK Element	4-4
4.4	Statement DEScription and Code LIBrary	4-4

CONTENTS (continued)

Chapter 5 - Installation and Dump of User Elements in the Controller

5.1	Getting Started	5-1
5.2	Down-Load	5-1
5.3	Pre-processing	5-3
5.3.1	Pre-processing PED Library	5-3
5.3.2	Pre-processing Library Code.....	5-4
5.4	User Elements in an Application Program.....	5-6
5.5	Installation Example.....	5-6
5.6	Dump and Load of Total Data Base and Library Programs.....	5-8
5.7	Dump and Load of Single Library	5-9
5.8	Total Library Installation.....	5-10

Chapter 6 - Change Handling and Maintenance

6.1	Change Handling.....	6-1
6.2	Performing Changes when the Library Programs are Available.....	6-2
6.3	Changing Libraries in a Dedicated Controller	6-3
6.4	Force PED Load.....	6-4
6.5	Administrative Spare Factor.....	6-6

Chapter 7 - Performance

7.1	Load Calculation	7-1
7.2	Memory Consumption	7-2
7.3	Library Size.....	7-3
7.4	Calculation Example	7-3

Chapter 8 - Test and Documentation

8.1	Test	8-1
8.2	Documentation	8-1

Chapter 9 - Error Codes and Messages

9.1	Message Type and Format	9-1
9.2	Error Codes	9-1
9.3	System Halt Codes	9-6

CONTENTS (continued)

Appendix A - System Messages

A.1	General.....	A-1
A.2	Command Monitor and Utilities (FAFA100).....	A-1
A.3	Statement Description Pre-processor (FAFA200)	A-5
A.4	Code Library Pre-processor (FAFA300).....	A-12
A.5	Common Pascal Routines (FAFB01).....	A-14

Appendix B - User Element Related Files in FCB Directory Structure

B.1	General.....	B-1
-----	--------------	-----

Appendix C - Group of Characters Applied in PED and TED Files

C.1	General.....	C-1
-----	--------------	-----

Chapter 1 Introduction

1.1 Before You Begin

The system software for Advant Controller 400 Series is based on one basic software module to which you add additional software modules.

The function User Defined PC Elements is a software module which enables you to simplify implementation and documentation of frequently used control solutions and adapt them to your own wishes. To improve the readability in this manual, we call the User Defined PC Elements **User Elements**. On the screen, the User Elements will appear as rectangles with input and output terminals.

You can design and combine a number of User Elements into **libraries** which you down-load into the controller. You can modify, add new elements to existing libraries, or create new libraries when required.

This manual is intended for all users of User Elements, and gives you guidelines on how to design your own PC elements. It contains information on how to start and how to perform various tasks related to configuration and maintenance of User Elements.

CAUTION

The controller will be set in configuration mode and stalled if you do not follow the procedure for creating User Elements correctly, for example try to run an application program with missing or insufficient Statement Description library (SDES) or code library (CLIB).

1.2 Equipment Requirements

1.2.1 Hardware

The function User Defined PC Elements is designed for the Advant Controller 410 and the Advant Controller 450.

1.2.2 Software

The function User Defined PC Elements is a software module. The module designation depends on which controller you use:

- QC01-UDP11*<8.0 and later versions> for Advant Controller 410
- QC02-UDP21*<8.0 and later versions> for Advant Controller 450
- QC07-UDP41*<1.0 and later versions> for Advant Controller 450

Additional load introduced by this module is approximately 0.1%.

1.2.3 Tools

We recommend that you use AdvaBuild Function Chart Editor 4.0 or later when defining the User Elements. Version 4.2 is used in the example in this manual.

AdvaBuild for Windows with corresponding On-line Builder for the software package below have to be used for all commands described in this manual.

Table 1-1. Corresponding Software

Software	Tools
QC01-UDP11 QC02-UDP21	AdvaBuild for Windows 1.2 AdvaBuild for Windows 1.3 AdvaBuild for Windows 1.4 AMPL Control Configuration 1.4
QC07-UDP41	AdvaBuild for windows 1.4 AMPL Control Configuration 1.4

1.3 Conventions

Table 1-2. Syntax Conventions

Convention	Description
boldface	In a syntax rule statement or example, a word in boldface represents a reserved, keyword or string
<i><variable></i>	Words in italics enclosed in angle brackets represents a variable or reference that must be replaced with the actual value.
DISPLAY	Word in display text font indicates system response or message. Related only to the AdvaBuild On-line Builder.

1.4 Related Documents

Table 1-3 lists documentation related to User Defined PC Elements

Table 1-3. Related Documentation

Title	Description
On-line Builder User's Guide	Describes how to install and setup the On-line Builder and the menu commands for Windows and HP-UX.
On-line Builder Reference Manual.	Describes all commands used in Advant Station 100 Engineering Station, Advant Station 500 Engineering Station and Advant Station 500 Operator Station. Command sets and working methods are described as well as command file handling aspects.

1.5 Terminology

Table 1-4. Terminology

Term	Description
AAB_FILE	DB element applied for loading of data in the controller's memory area
AC 410/450	Advant Controller 410 and Advant Controller 450
CLIB	Code LIBrary. Internal system definition of AMPL library program installed in a controller
Compile	Conversion of file from text format to binary format readable by FCB
Controller	Advant Controller 410 or Advant Controller 450
Control solution	Definition of control function to be implemented as User Element. Contains a sequence of conditions and functional relationships describing how to control an object or a process
DB	Controller Data Base
DB element	Data Base element
Down-load	An operation transferring a loadable format from an engineering tool to the controller (both in configuration and operational mode). Up-load is the opposite

Table 1-4. Terminology (Continued)

Term	Description
Engineering tool	Used in this manual as common designation for ABB software tools and commands needed for application programming, documentation and on-line configuration. Refers normally to the AdvaBuild family of functions
FCB	AdvaBuild Function Chart Builder. The Function Block tool for AMPL programming and documentation of Advant Controller programs
FPS_WORK	Name of the AAB_FILE element, applied as user's command interface and temporary storage area for down-loaded PED library
Interface element	Supporting PC element applied only in AMPL library programs. Created automatically in the controller during the pre-processing. To be recognized and displayed in FCB, you must create PED library for interface elements and compile it.
LF	Logical File in the Advant Controller data base
Library program Library (source) code	Application program containing an AMPL source code for at least one User Element. Short name library code is also applied
LR	Logical Record inside the Advant Controller data base. Corresponds to one DB element
MDB	Modify DB element - AdvaBuild On-line Builder command
On-line Builder	The AdvaBuild tool for on-line programing and configuration of the controllers (both in configuration and operational mode)
PC elements, standard	Designation standard (or ordinary) PC element is used to distinguish User Elements from PC elements supplied with the controllers
PED library	PC Element Description library. Contains element description for one or several User Elements
PED, PE, PEL	Extension for files containing PED library. PED = text (ASCII) format. PEL = binary format, readable by FCB. PE = same content as PED but located in a controller environment (SRCE directory).
Pre-process	Operation similar to a software compilation. Down-loaded PED library or AMPL library source code are converted to User Elements that can be recognized and executed inside the controller. See also CLIB and SDES

Table 1-4. Terminology (Continued)

Term	Description
Pre-processed library file	File created by On-line Builder's command DUMPFIL. Contains dump of a single SDES or CLIB in a format that can be transferred and loaded in other controllers.
<p.v/r>	Product release.version/revision
SDES	Statement Description library. Internal system definition of PED libraries installed in the controller. Automatically created during the pre-processing.
Source code	AMPL program or data base contents in ASCII format or format readable by the controller
Target Description	Description of software options (PED libraries) intended for use only with a particular controller (target) type. Used only by FCB to identify and display various options for different controllers.
TED, TEL files	Extension for files containing target description in different formats. TED = text (ASCII) format. TEL = binary format, readable by FCB.
User Elements	User Defined PC Elements

1.6 Release History

Table 1-5 lists the development of User Defined PC Elements.

Table 1-5. Release History

Version	Description
1.0	This is the initial User Defined PC Elements release.

Chapter 2 Installation

2.1 Introduction

A user's control solution designed and implemented as a User Element will appear in application programs along with ordinary PC elements. This chapter gives you an introduction to the concept of User Elements and an overview of the working flow to follow when creating your own User Elements. A schematic overview is shown in [Figure 2-6](#).

2.2 Control Solution

Assume that user specific control design for valve operation contains logic as in the figure below.

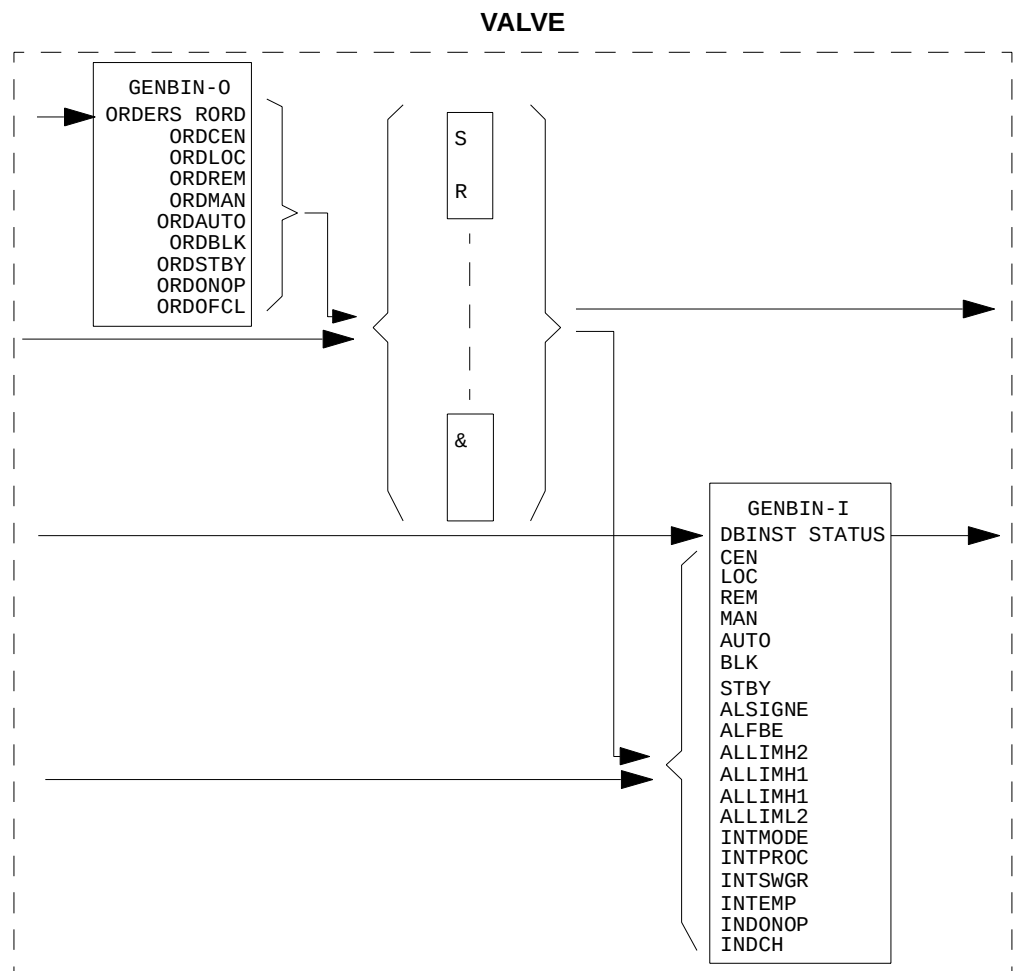


Figure 2-1. User's VALVE Control Solution

Functional unit GENBIN is used in the example for operator control and indication.

This control solution will be available to the application designer as a User Element named VALVE with the appearance in the application programs as shown in Figure 2-2. Input values to the control solution are entered through connections to input terminals of the User Element. Results of the control solution's processing are displayed on output terminals. An overview of the transformation from the illustrated control solution to an executable User Element will briefly be explained in the succeeding sections of this chapter.

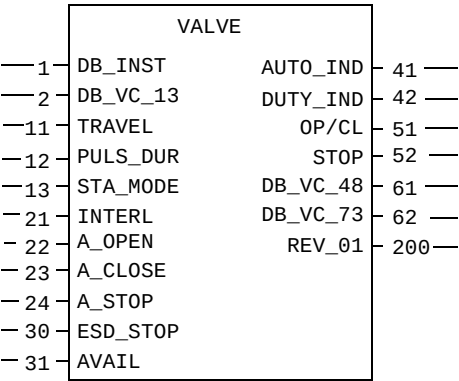


Figure 2-2. User Element Valve

2.3 Creation of User Elements in FCB

2.3.1 Definitions

User Element Description

The User Element's name and input/output terminals with their names, numbers and data types must be defined in order to be identified, interpreted and displayed. This definition is called User Element description.

PED Library

PC Element Description (PED) library is a collection of descriptions for User Elements related to some project, application area or product

Interface Elements

The control logic within a User Element is terminated in supporting PC elements named Interface elements. These elements are applicable only when you create the control solution for a User Element. Depending on the selected call parameters, an interface element appears as an input or output element as indicated in Figure 2-3. The interface input element has only output terminals identical to the corresponding User Element's input terminals. The interface output element has only input terminals identical to the corresponding User Element's output terminals. The User Element's control solution, resides between these two elements (see Figure 2-4).

The interface element's call name is the same as the name of the corresponding User Element, but with an added character "I".

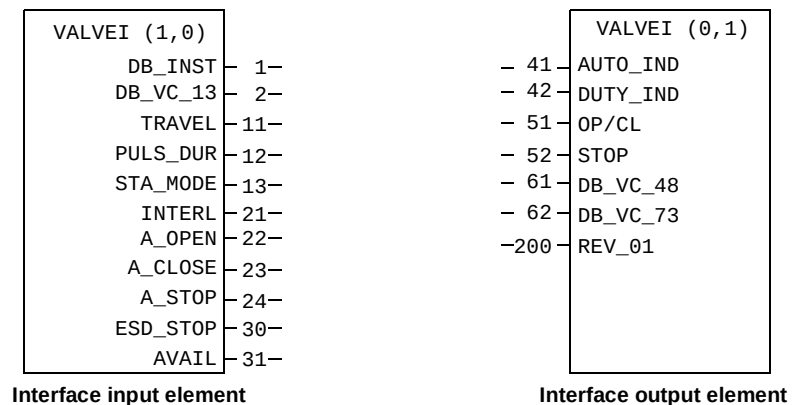


Figure 2-3. Input and Output Interface Elements for the User Element VALVE

Target Description

When selecting a node type in the Application Builder (APB) you have the possibility to select one or several PC element libraries delivered with the tool. These libraries, also called options, appear in the APB window "Available Options". One or several libraries are available for each controller (node) type.

In addition to creating User Element descriptions you have to define for which node type your PED libraries are designed for and which standard PC element libraries are needed. This definition is called target description.

Target description is needed only when you work with FCB.

Library Program

A PC program with control solutions for User Elements described in a particular PED library is called library PC program (library program). This program is an AMPL program containing standard PC elements and interface elements. To avoid a confusion in terminology the PC program applying User Elements will be called **application program** (contrary to a library program).

Library Source Code

The library program's content in ASCII format or format readable by the controller is called a library source code.

2.3.2 User and Interface Elements in FCB

Before you start creating a library program with your control solutions in FCB, you have to install the interface elements so they are available in the FCB graphical editor. At the same time, install the User Elements.

Create PED Library

First of all, define your User Elements by creating a PED library file with a text editor. This file contains a description for at least one User Element. Normally, it defines several User Elements related to some application area, project or product. Details on how to build up a PED library file are given in [Section 3.1, User Element Description](#)

Create PED Library for Interface Elements

This library is automatically created by applying a simple command delivered with your AdvaBuild. A description for one interface element is created for each User Element and located in a text file in the same order as respective User Elements in the PED library file. You need this library only when you work with FCB.

Create Target Description File

A definition of the node type your PED libraries are designed for, and dependences to standard PC element libraries is done by creating a target description file. Details on how to create target description file are given in [Section 3.4.2, Creating a TED File](#).

Compile PED Libraries to Binary Representation

Installation to FCB is done by compiling both PED library files to binary format readable by the FCB. This is done by applying AdvaBuild's Setup Add-Ons function. Target description file must be compiled along with the PED library files. After compilation, User and interface elements can be interpreted by FCB and displayed in form of rectangles with input and output terminals. The interface elements are available for use in a library program with your control solutions. The User Elements are also available and can be inserted in an application program along with the PC elements.

2.4 Creation of Library Program and Source Code

When you have performed the tasks described in the previous section, you can start designing the library program with your control solutions.

The library program contains control design of at least one User Element. The library program must have a strict program structure: start with a PCPGM element immediately followed by the first control module. The first element following the CONTRM header must always be the interface element, with call parameters defining it as an input element. The last element in the CONTRM must always be an interface element, with call parameters defining it as an output element. One control module contains the internal control design for only one User Element. [Figure 2-4](#) shows the control solution illustrated in Section 2.2 when it is implemented in FCB, as part of a library program.

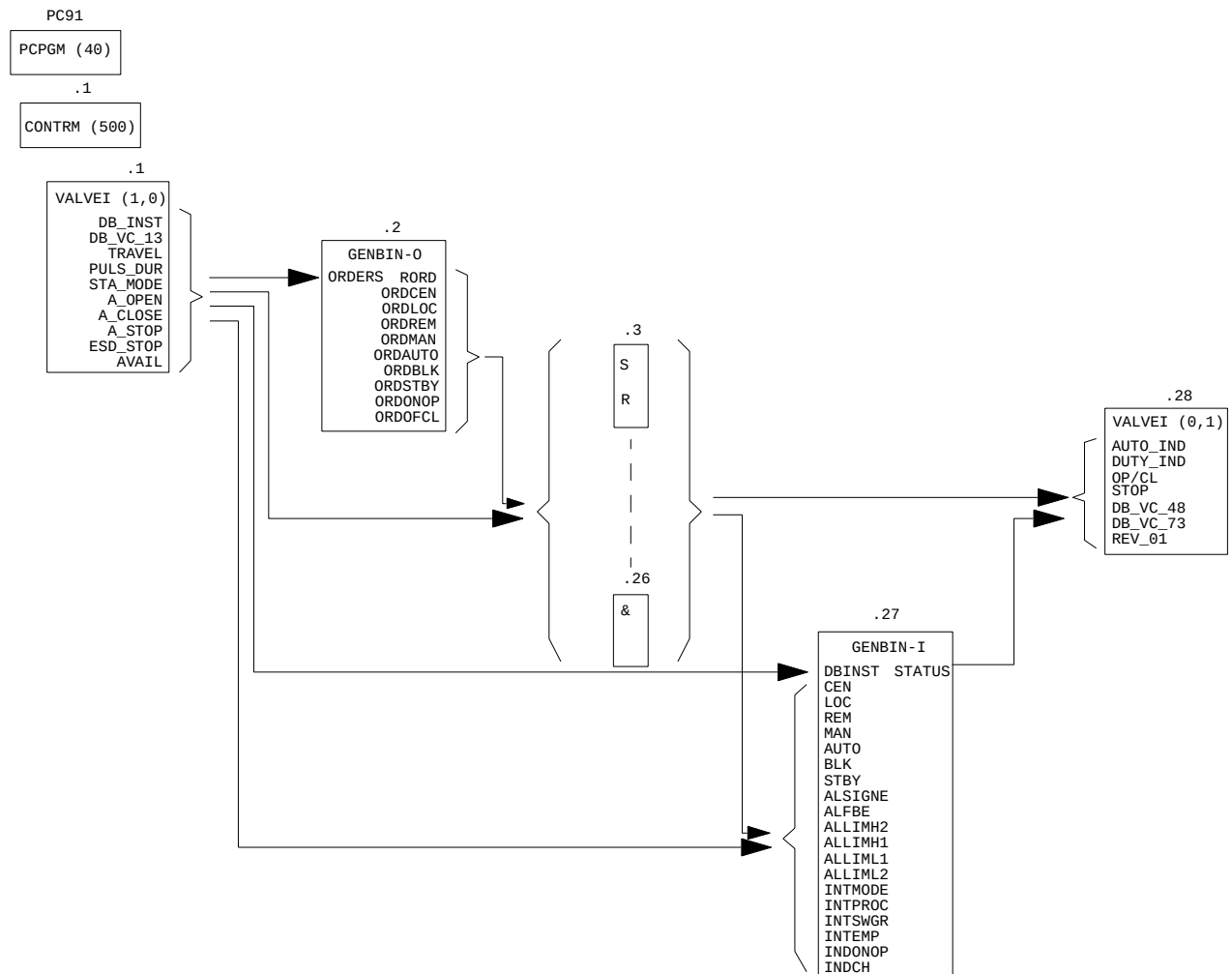


Figure 2-4. Part of Library Program containing Control Design for User Element VALVE.

When the library program is completed, the source code generated by FCB is ready for download in the controller.

2.5 User Elements in the Controller

2.5.1 Definitions

Installation of Libraries

The term installation is a common designation for two operations: down-load and pre-processing. The installation of one library means installation of a PED library and the related library source code. The installation of libraries in the controller is described in detail in [Chapter 5, Installation and Dump of User Elements in the Controller](#).

AAB_FILE Element

The DB element AAB_FILE is applied when you want to transfer data, other than AMPL source code, to a storage data area in the controller. During the installation in the controller, two AAB_FILE elements are created automatically for each library you install. Details on AAB_FILE elements are given in [Chapter 4, User Elements in the Controller](#).

AAB_FILE Element FPS_WORK

Prior to installation you must create a AAB_FILE element with name FPS_WORK. This is a reserved name applied only when working with User Elements. By creation of this element you actually reserve a storage data area in the controller, used during the installation.

Down-load

Down-load is a process of transferring data from an engineering tool to a data area in the controller.

Pre-processing

Pre-processing is an internal conversion of down-loaded data to User Elements executable by the controller.

2.5.2 Installation in the Controller

Assuming your PED library and the related library source code are created and available to the On-line Builder, the installation can take place. The working flow is as follows:

1. Dimension the data base for AAB_FILE elements

The Number of AAB_FILE elements you must dimension, depends on the number of libraries to be installed. You need two AAB_FILE elements for each library, in addition to the FPS_WORK element. Dimension your data base for a number of AAB_FILE elements as well as the storage data area for AAB_FILE's data. For details, see [Chapter 5, Installation and Dump of User Elements in the Controller](#).

2. Create FPS_WORK

Prior to installation, you create the FPS_WORK element. This DB element is used as a temporary storage data area for PED library's data before the pre-processing. After a successful pre-processing, the area content remain unchanged until you decide to down-load a new library. The area content is then overwritten by the new data loaded. You use this element also for ordering the operations related to installation and maintenance. For details see [Chapter 5, Installation and Dump of User Elements in the Controller](#) and [Chapter 6, Change Handling and Maintenance](#).

3. Down-load a PED library

You perform a down-load of a PED library by applying the On-line Builder's command LOADFILE. The PED data are transferred to a temporary storage data area reserved by creation of FPS_WORK.

4. Pre-process the PED Library

PED data residing in the temporary storage area is a target for pre-processing. You request the pre-processing by entering a command number on FPS_WORK's terminal VERSION. The operation is started when you enter value = 1 on FPS_WORK's terminal RW_PROT.

The interface elements are automatically created during the pre-processing. Both your User Elements and the related interface elements are now available in the controller, but they can still not be executed.

5. Down-load the Library Code

Down-load of a library source code is done in ordinary way as any other AMPL source code. The On-line Builder's command TRPCS is applied.

6. Pre-process the Library Code

Pre-processing of the library source code is an internal transformation of down-loaded library source code to executable User Elements. You request the pre-processing in the same way as described for the PED library.

7. Next Library

You repeat the installation process for the next library starting with down-load of the next PED library.

2.6 Applying User Elements

After installation of all libraries, you can down-load the application programs with User Elements, created in FCB. As soon as you disable the build mode your application programs are running in the controller. You can insert new or delete existing User Elements and perform any other operation that is allowed for ordinary PC elements.

Figure 2-5 shows part of a typical structure of a library program and how it relates to an application PC program with the User Elements concerned.

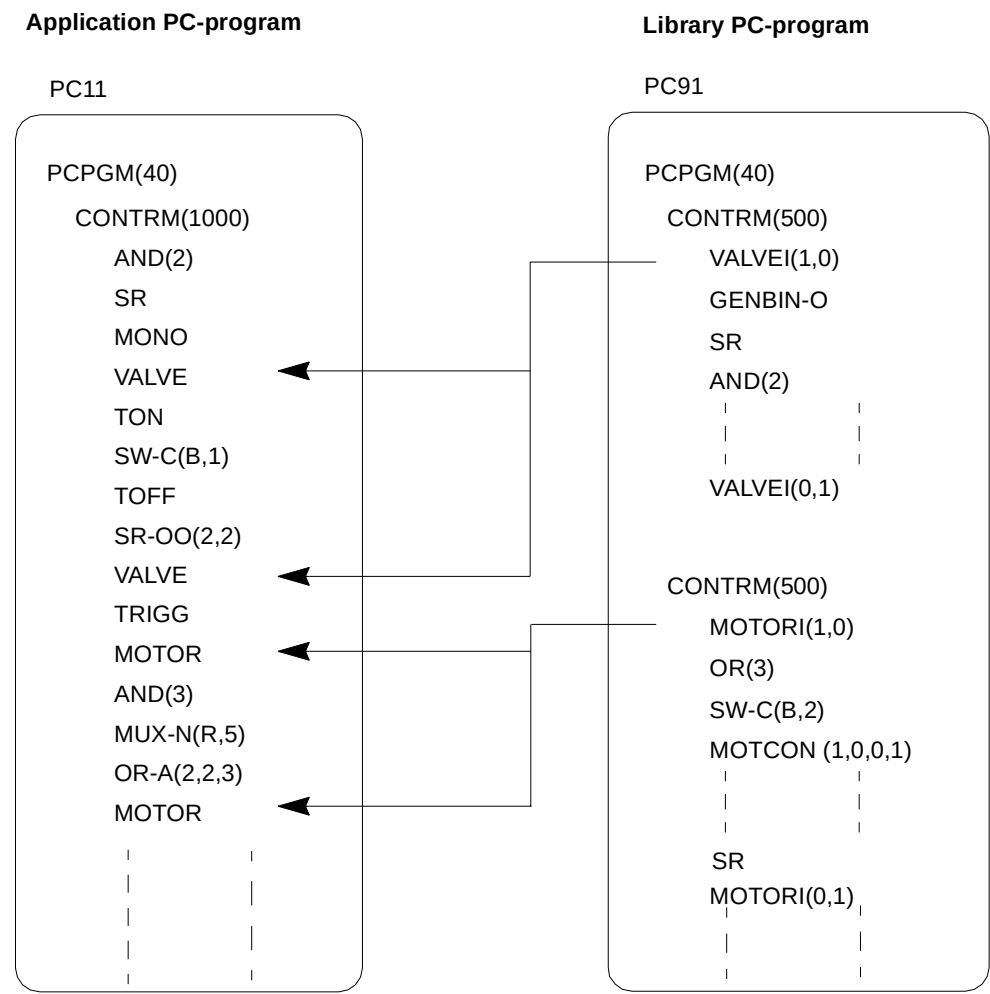


Figure 2-5. Example of an Application/Library PC Program Structure

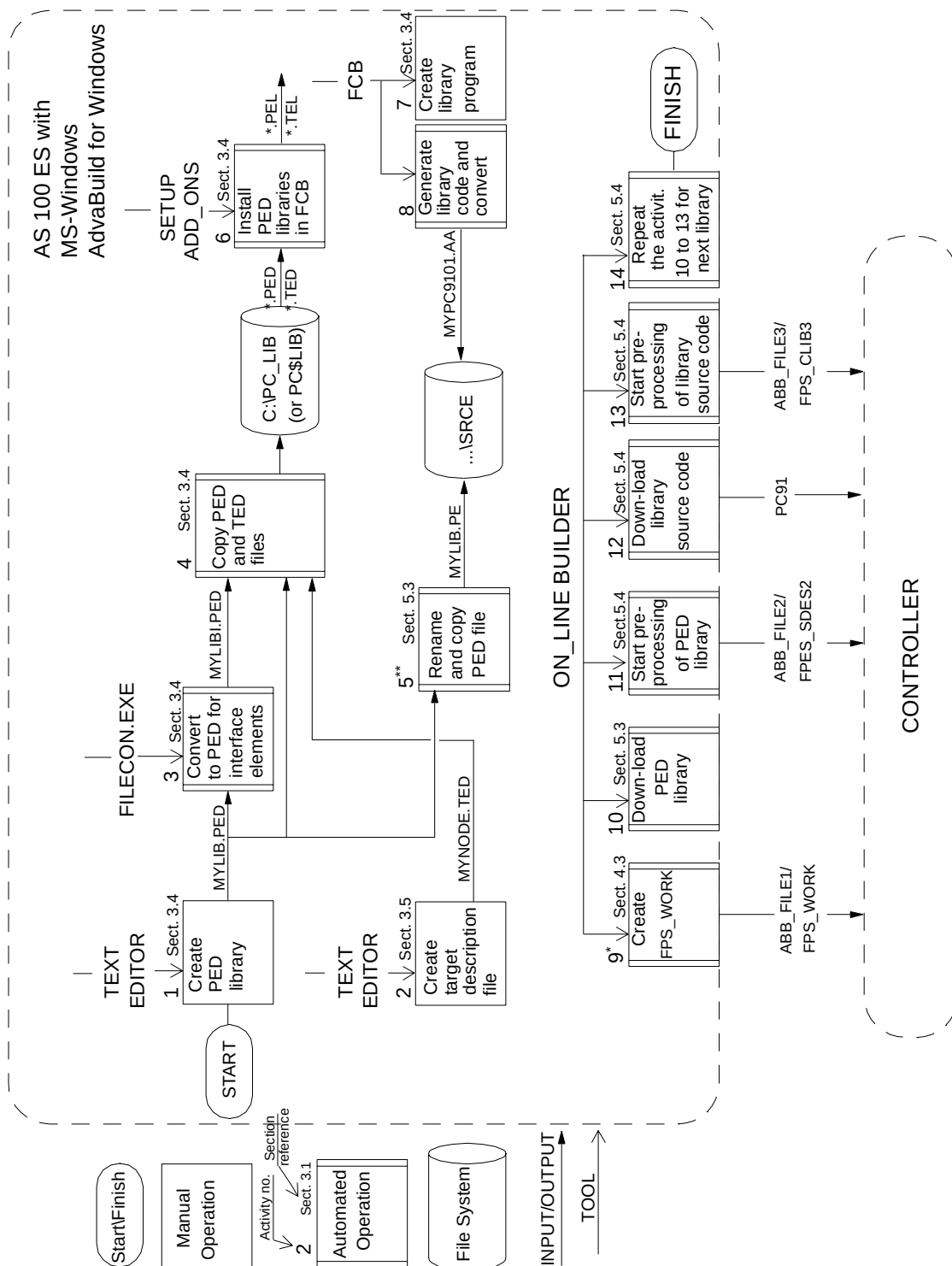


Figure 2-6. Producing Libraries in FCB, Down-load and Pre-processing. Schematic Overview.

Chapter 3 Configuration/Application Building

3.1 User Element Description

Definition of a User Element's call name and input/output terminals with their names, numbers and data types is done by creating a text description file with a text editor.

NOTE

If you use a common word processor save your file as "text only". Otherwise, a control characters automatically inserted by the processor might cause you problems.

This file, called PC element description file (extension: PED), contains description for at least one User Element. Descriptions for all User Elements within a particular project, application area, or related to some product, are normally located in one PED file. A typical example of an element description for a User Element is given below.

The content of MYLIB.PED:

```
LIBTYPE := "2";
LIBNAME := "MY-LIB";
LIBVERSION := "1.1";

pc_element VALVE

COMPUTINGTIME := 285;
PC_STATEMENT := 948;
LOCAL_DATA := 380;

NONEXECUTABLE
4 FUNCM BLOCK CONTRM SLAVEM STEP;
LEVEL 1 "VALVE" FIXWIDTH 18;
terminal 1 name "DB_INST" DBREF type IN I;
terminal 2 name "DB_VC_13" type IN A * 32 ;
terminal 11 name "TRAVEL" type IN TR ;
terminal 12 name "PULS_DUR" BLANKLINE type IN T ;
terminal 13 name "STA_MODE" DYNAMIC INVERTABLE type IN B ;
terminal 21 name "INTERL" ADJUST INVERTABLE type IN B ;
terminal 22 name "A_OPEN" INVERTABLE type IN B ;
terminal 23 name "A_CLOSE" INVERTABLE type IN B ;
terminal 24 name "A_STOP" INVERTABLE type IN B ;
terminal 30 name "ESD_STOP" INVERTABLE type IN B ;
terminal 31 name "AVAIL" INVERTABLE type IN B ;
terminal 41 name "AUTO_IND" ADJUST type OUT B ;
terminal 42 name "DUTY_IND" type OUT B ; (* always in duty *)
terminal 51 name "OP/CL" type OUT B ;
terminal 52 name "STOP" type OUT B ;
terminal 61 name "DB_VC_48" type OUT G 10 I ;
terminal 62 name "DB_VC_73" SLINSTANCE type OUT A * 12 ;
terminal 200 name "REV_01" type OUT R ;
end VALVE;
```

Description for the next element in a library continues in the same way beginning with the line:

pc_element <User Element call name>

and ending with line:

end <User Element call name>.

The example together with the rules given below describes how to create your own PED library files.

- The **Library type** is always 2.
- The **Library name**, defined by the user, can have max. 8 characters. The allowed characters are given in [Appendix C, Group of Characters Applied in PED and TED Files](#)
- The **Library version**, defined by the user, is an alphanumeric designation of library version and revision. The allowed characters are given in [Appendix C, Group of Characters Applied in PED and TED Files](#).
- The **pc_element** name, with max. 7 characters, must be different from any standard PC element name (AND, OR, etc.). The allowed characters are given in [Appendix C, Group of Characters Applied in PED and TED Files](#).
A User Element must have an unique name within the library and the controller it is intended to be installed in.
- The value of execution time is entered in microseconds in the **COMPUTINGTIME** line. Execution time calculation is explained in [Section 7.1, Load Calculation](#).
- The values of memory consumption for **PC_STATEMENT** and **LOCAL_DATA** are to be entered in bytes. Memory consumption calculation is explained in [Section 7.2, Memory Consumption](#).
- The element type is always **NONEXUTABLE**, which means that no call parameters defining execution time are applicable. Call parameters are not applicable at all.
- The PC class is always **4**, indicating a group of PC elements that can be inserted on the same level as the User Element concerned. Members of class 4 are all standard PC elements besides structure elements. Exception: BLOCK element belongs to class 4.
- A User Element may be inserted on lower level to the structure elements stated in the example. It may not be inserted on lower level immediate after PCPGM, SEQ and MASTER.
- The terminal **LEVEL** is always 1. The name entered in the **LEVEL** line shall always be the User Element's name.
- The width of the User Element rectangle may be modified by entering an integer value preceded by the keyword **FIXWIDTH**. This value represents a number of character places across the rectangle width. Standard size is 17 characters which corresponds to two terminals of 8 characters each plus one character in between. This keyword shall be applied only when you want to alter the standard width size.

- The **terminal** number is an integer in range 1 to 255. Input and output terminals, declared with words **IN** and **OUT**, can be specified in random succession. During the installation in the controller, the terminals will be sorted in a consecutive input and output section. One output terminal must be dedicated to revision indication. Name of the terminal shall be written as REV_<rev.design.>. The revision designation <rev.design.> is an alphanumeric string intended to provide traceability in case of functional modification of an existing element.
Function parameter terminal can not be defined as User Element terminal. On standard PC elements a function parameter is identified by a letter 'F' followed by a number between 1 and 99.
- The terminal **name** can have max. 8 characters. A list of allowed characters is given in [Appendix C, Group of Characters Applied in PED and TED Files](#). Two or more terminals with the same names are not allowed.
- Valid terminal type strings are **B, I, IL, R, T, TR, A, and G**. Number of bytes in terminals of type array and string must be within the range 1 to 255. Number of group items in terminals of type group must not exceed 255.
- The user **comments** can be added anywhere provided they are enclosed in brackets and asterisk as shown in the example.
- Some key words, so called **descriptors**, that can be interpreted by the AdvaBuild Function Chart Builder (FCB) should also be included. The descriptors used in the example are **DBREF, INVERTABLE, DYNAMIC** and **ADJUST**. For more details on this topic, see [Section 3.2](#).

Syntax Rules

- All characters with ASCII code less than 32 (control characters) and space are considered to be de-limiters.
- Multiple space is interpreted as a single space.
- The keywords are: LIBTYPE, LIBNAME, LIBVERSION, pc_element, COMPUTINGTIME, PC_STATEMENT, LOCAL_DATA, NONEXECUTABLE, LEVEL, FIXWIDTH, terminal, name, type, end, IN, OUT, I, IL, B, T, TR, R, A, S and G. The terminal descriptors as well as the structure PC element names are also keywords.
- The keywords must be preceded and succeeded with at least one space or another de-limiter character, in order to be interpreted as a keyword.
- The semicolon marks the end of a line. A line is therefor interpreted as a sequence of characters ending with “;”.
- All words in a PED file are case sensitive, except from user comments and words in quotation marks.
- The User Element name in **pc_element** line and **end** line must be written in uppercase.
Comment can be written anywhere, providing that it starts with “(“ and ends with “)“.

3.2 Terminal Descriptors

Terminal descriptor is keyword assigned to a terminal for the purpose to alter a function associated to it or the element's layout in the FCB graphical editor. The terminal descriptor is entered in a PED file between the terminal name and the keyword **type**. Figure 3-1 shows how some of the descriptors affect the element appearance in FCB. There are six descriptors of interest when working with User Elements:

DBREF indicates a terminal intended for connection to DB property identifying a general object instance, for example, GENBIN. The object name entered in a FCB dialog does not need to be followed by property number or name.

INVERTABLE is only valid for boolean input terminals. If not present, FCB rejects any attempt to invert an input terminal.

DYNAMIC is only valid for boolean input terminals. Applied to indicate triggering inputs. Displayed in FCB with symbol ">" at the terminal name as shown in Figure 3-1

ADJUST is used to align horizontally an output terminal with a referring input terminal (in the example, AUTO_IND is aligned with INTERL).

SLINSTANCE solid line instance. A solid line appears under the terminal containing this descriptor (AVAIL in the example).

BLANKLINE blank (empty) line. A blank line appears under the terminal containing this descriptor (PULS_DUR in the example).

The first two descriptors affects a function associated to a terminal. The other descriptors affects the graphical appearance.

Use of graphical terminal descriptors help you to customize the graphical appearance of User Elements in the FCB. The following rules are to be applied when inserting terminal descriptors in your PED file:

- Insertion order - when you apply more than one graphical descriptor in the same terminal line, place DYNAMIC nearest to the terminal name and INVERTABLE nearest to the keyword **type**. The other are to be placed in between.
- Blocks of terminals - when you apply ADJUST, SLINSTANCE and BLANKLINE, your input and output terminals must be grouped in a block of terminals numerated in consecutive order.

For instance, if one block consist of terminals numbered 1 to 20, the next block must begin with terminal number 21 or higher. None of the vacant numbers within the sequence 1-20 can be used in an other block of terminals.

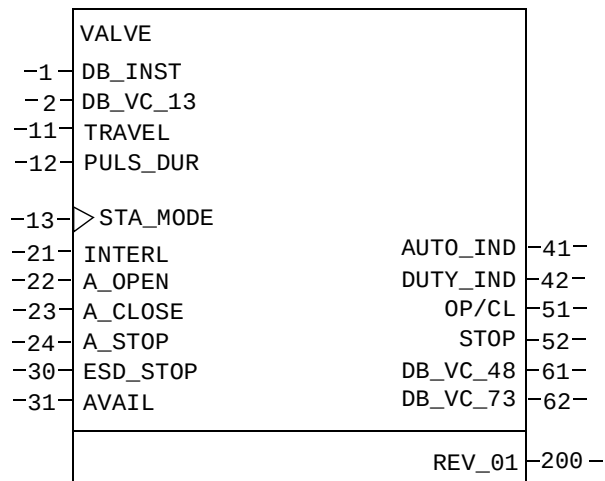


Figure 3-1. User Element VALVE as it appear in FCB

NOTE

DBREF is the only descriptor that is interpreted by the controller. The AdvaBuild On-line Builder does not pay any attention to the other terminal descriptors. A User Element does **not** appear in the On-line Builder window as shown in [Figure 3-1](#).

3.3 Interface Element Description

Definition for interface elements is created by applying file conversion function:

FILECON.EXE.

If you use AMPL Control Configuration 1.4 you will find this function in FCB/<release>/BIN directory. AdvaBuild users will find it located in On-line Builder's directory structure. If you have installed AdvaBuild without On-line Builder, then:

- Insert the AdvaBuild installation diskette No. 10 in your local drive and copy the compressed file FILECON.EX_ to Application Builder's directory.
- Enter the APB directory on DOS level and execute the DOS command EXPAND. Delete the compressed file after a successful expansion.

Execution is done in DOS or MS-Windows environment. The syntax rule is:

FILECON <ped file>.PED <ped file>I.PED

The resulting file shall have the same name as the source file, but with an added character I.

The converted file contains declaration of call parameters and description of input and output interface elements. The example from [Section 3.1](#) will after the conversion look as shown below. The library name should be changed manually, by the reason that will be explained in [Section 3.4.2](#).

The content of MYLIBI.PED :

```
LIBTYPE := "2";
LIBNAME := "MY-LIB-INTERFACE";
LIBVERSION := "1.1";
pc_element VALVEI ([C1 74,C2 74:0..1 := 0])

COMPUTINGTIME := 0 + 4*C1 + 6*C2;
PC_STATEMENT   := 8 + 2*C1 *11+ 2*C2 *7;
LOCAL_DATA     := 0 + 62*C1;

NONEXECUTABLE
4 FUNCM BLOCK CONTRM SLAVEM STEP;
LEVEL 1 "VALVE" FIXWIDTH 18;
if C1 then
terminal 1 name "DB_INST" DBREF type OUT I;
terminal 2 name "DB_VC_13" type OUT A * 32 ;
terminal 11 name "TRAVEL" type OUT TR ;
terminal 12 name "PULS_DUR" BLANKLINE type OUT T ;
terminal 13 name "STA_MODE" DYNAMIC INVERTABLE type OUT B ;
terminal 21 name "INTERL" ADJUST INVERTABLE type OUT B ;
terminal 22 name "A_OPEN" INVERTABLE type OUT B ;
terminal 23 name "A_CLOSE" INVERTABLE type OUT B ;
terminal 24 name "A_STOP" INVERTABLE type OUT B ;
terminal 30 name "ESD_STOP" INVERTABLE type OUT B ;
terminal 31 name "AVAIL" INVERTABLE type OUT B ;
end if;

if C2 then
terminal 41 name "AUTO_IND" ADJUST type IN B ;
terminal 42 name "DUTY_IND" type IN B ; (* always in duty *)
terminal 51 name "OP/CL" type IN B ;
terminal 52 name "STOP" type IN B ;
terminal 61 name "DB_VC_48" type IN G 10 I ;
terminal 62 name "DB_VC_73" SLINSTANCE type IN A * 12 ;
terminal 200 name "REV_01" type IN R ;
end if;

relations
if C1+C2 <> 1 then error;
end VALVEI;
```

NOTE

Computing time and memory consumption must be added manually in the PED file for interface elements. The FILECON command does not generate these calculation automatically. Be aware that the calculation data entered manually are overwritten each time you apply FILECON.EXE.

PED library for interface elements is needed only for installation in FCB as it will be explained in [Section 3.5](#) . This library is not needed in the controller at all.

3.4 Target Description (TED)

3.4.1 Definition

Basic and Optional Software

Standard PC libraries are delivered as basic and optional software modules suitable for installation in the controller. The basic PC libraries are always included in the controller. Optional PC libraries are ordered separately and installed, either by the supplier or the user.

The basic and optional PC libraries available for different controllers are displayed in the APB's windows "New Node" and "Modify Node". When you define a new or modify an existing node (controller) you select only options that are present in your controller as software modules (delivered on PROM card).

User Defined Options

Libraries containing User Elements are also considered as options. However, they are created and installed by the user, in a controller equipped with the User Element software module.

The PC elements applied in your library programs may reside in various PC libraries. If these libraries are not present in the controller (optional software module not installed), the library source code concerned can not be down-loaded.

The term option is generally used in this Chapter to designate one optional (not basic) library, no matter if it is created by the user or ordered as optional software module.

Target Description

Definition of basic and optional PC libraries available for a particular controller and its dependences is made by creating a Target Description file (extension: TED). These files are present in the FCB for both the Advant Controller 410 and the Advant Controller 450.

In order to get your libraries present in FCB and displayed in the APB's "Available Options" window, you have to create a target description file and compile it as it will be described in [Section 3.5](#).

The following is defined in a TED file:

- Controller type (AC 410 or AC 450), and which version your libraries are designed for
- Which options your libraries depends on
- The names of your PED library files created for the actual controller.

You must create one TED file for each controller type (and version) in case your libraries are intended to be used in more than one controller type.

NOTE

The target description file is not needed at all in the controller. It defines relations between options and node types needed when you work with Application Builder and Function Chart Builder.

3.4.2 Creating a TED File

An example of a target description is given below. Words written in **bold** are keywords and must be written in lower case.

The content of UDNODE.TED:

```
syntax 1.1;

modify target "AC 410" version 1.0;
add
  option USER-ELEMENTS version 1.1;
  comment "MY udPCe library";
  supports
    pc_element library MY-USER-elements version 1.1/0 stored in MYLIB;
    requires option "QC01-LIB11" version 7.0/0;
    option "QC01-LIB12" version 7.0/0;
  end_option;

  option INTERFACE-ELEMENTS version 1.1;
  comment "MY udPCe library";
  supports
    pc_element library MY-INTERFACE-elements version 1.1/0 stored in MYLIBI;
    requires option "QC01-LIB11" version 7.0/0;
    option "QC01-LIB12" version 7.0/0;
  end_option;
```

The description for the next option continues in the same way, beginning with line **option** <option name> and ending with line **end_option**.

The example above together with the rules and limitations, given in this section, help you to create your own TED files. The explanatory paragraphs below refer to specific parts of a TED file.

Syntax identifier is 1.1.

As your library is not a standard option, the node type concerned must be modified in order to accept user defined options. This is indicated in the **modify target** line. Node type in this line, enclosed in quotation marks, is written with space between letters and number: "AC 410" or "AC 450". The exact name of all available node types is found in FUNCTION.INI file located in ...\\Advabld\\FCB\\<release dir>\\BIN.

The name of the **option**, with max. 18 characters, must be unique. This name appears in the APB "Available Options" window, as shown in [Figure 3-2](#). A list of allowed characters is given in [Appendix A, System Messages](#).

Valid **version** for an option is an alphanumeric designation that should be written in the form <p.v/r>. The exact name of standard options and their versions is found in FUNCTION.INI file located in ...\\Advabld\\FCB\\<release dir>\\BIN.

The user **comment** with max. 30 characters, must be set in quotation marks. A list of allowed characters is given in [Appendix C, Group of Characters Applied in PED and TED Files](#).

User's library name is stated in **pc_element library** line with max. 20 characters. It should refer to the LIBNAME declared in the related PED library. A list of allowed characters is given in [Appendix C, Group of Characters Applied in PED and TED Files](#).

User's library **version** must be specified exactly as in the PED library concerned.

Several libraries can be declared within an **option** provided that their names and content are unique.

If you, for example, want both the User Elements and the related interface elements to be parts of the same option, the **pc_element library** declaration in the example will look like:

supports

```
pc_element library MY-USER-elements version 1.1/0 stored in MYLIB;  
pc_element library MY-INTERFACE-elements version 1.1/0 stored in MYLIBI;  
requires option "QC01-LIB11" version 7.0/0;  
option "QC01-LIB12" version 7.0/0;
```

In that case, both libraries will be available in the controller by selecting the option USER-ELEMENTS. One user defined library can be a part of several options. If so, do not select options containing the same libraries (User Elements). Otherwise, the text "Element already in library" appears when you try to enter the actual node in FCB.

PED library file designation, with max. 8 characters, is preceded by the keyword **stored in**.

Elements within the User Element library can not be executed if options declared in the **requires option** and subordinate **option** lines are not present in the controller. When a User Element library is selected in APB "Available Options" window, all required options are automatically selected (see the right lower box in [Figure 3-2](#)). If a required option is a user defined, you must indicate where it is stored. If your option, for example, requires option named "Some Library" stored in somefile.ped, the **requires option** declaration in the example will be:

supports

```
pc_element library MY-USER-elements version 1.1/0 stored in MYLIB;  
requires option "Some Library" version 7.0/0 stored in SOMEFILE;
```

or

supports

```
pc_element library MY-USER-elements version 1.1/0 stored in MYLIB;  
pc_element library MY-INTERFACE-elements version 1.1/0 stored in MYLIBI;  
requires option "Some Library" version 7.0/0 stored in SOMEFILE
```

Node Name: UDNODE		Node Type: AC 410 1.0			
Net, Node: 1,1		Bus, Station: 		Base Software: QC01-BAS11 7.0/0	
Comment: 					
Available Options: INTERFACE-ELEMENTS 1.1 QC01-LIB12 7.0/0		 		Selected Options: QC01-LIB11 7.0/0 USER-ELEMENTS 1.1	

Figure 3-2. User Defined Options available in FCB along with the Standard Libraries

We recommend you to begin the name of a TED file with UD to indicate options with User Defined Elements. For instance, file named UDAC4101.TED contains options for target type Advant Controller 1.0. Names of the standard TED files delivered with the AdvaBuild installation diskettes are reserved.

3.5 Installing PED Libraries in FCB

Installation of PED libraries in FCB is performed by compilation of PED and TED files from text to a binary format, readable by FCB. For this purpose the AdvaBuild's Setup Add-Ons function is applied. The resulting files with extension PEL, PTL and TEL respectively are located in the FCB release directory:

- The *.PEL and *.PTL files in ...*<release dir>*\BIN\PC_LIB
- The *.TEL files in...*<release dir>*\BIN

NOTE

Before you install any additional options, copy FUNCTION.INI file, found in the release directory \BIN to a file named FUNCTION.001,

Setup Add-Ons search only in C:\PC_LIB or the directory you define by utilizing the DOS environment variable PC\$LIB. All PED and TED files found will be target for installation. In order to avoid warnings of type "Library already exist" the already installed PED files should be removed from the C:\PC_LIB (or PC\$LIB) directory. Copy your PED and TED files to this directory and proceed as follows.

- Start the Setup Add-Ons from the application icon
- Select "Compile only" followed by "Continue".

During the installation the name of PED files are continuously displayed in installation order. Otherwise, one or several error messages appear. Most of the messages are related to syntax errors.

If an error in a TED file is detected, a list file with the same name as the TED file concerned but with extension LIS is generated instead of TEL. Examine this file with a text editor, look for **** string and follow the hints.

If an error in a PED file is detected, a list file with the same name as the PED file concerned but with extension LST is generated instead of PEL. Enter this file with text editor, look for **** string and follow the hints.

After a successful installation, your libraries are available as options that can be selected in the APB's "Available Options" window along with the standard options (see [Figure 3-2](#)).

Attempt to install a TED file with name identical to an existing TEL file, present in the FCB release directory, will be rejected unless you check the "Replace destination file" box at compilation start.

The same is valid for a PED file if its PEL counterpart is already present in FCB.

An attempt to enter a node in FCB with PED libraries defined but not present in the release directory, will be rejected. A message "can not load PC element libraries" appears.

NOTE

Do not remove or delete the standard TEL and PEL files located in the FCB release directory. Be careful when deleting your TEL and PEL files.

FCB must be closed when you apply Setup Add-Ons.

User defined options can be composed of existing libraries stored as PEL and PTL files. In that case you need to create and compile only a TED file provided the PEL and PTL files concerned are copied to the FCB directory ...*<release dir>*\BIN\PC_LIB.

If you make changes in an installed PED library, you may compile only the PED file provided its name and version is kept as declared in the affected TED file. Compilation of a TED file is needed when you decide to change a PED file name or upgrade options version.

Assume you have designed an application program with User Elements that belong to a particular PED library. If some of the User Elements applied is to be modified, for instance a new terminal added, you make the modification in the PED file and perform re-compilation. When you start a new FCB session and inspect your application program, you will find out that the modification you made is not enforced. In order to get it effective, you have to initialize your node and backtranslate the application program(s).

When too many options appear in the APB “Available Options” window, remove those that are currently not in use. To do so, follow these steps:

1. Remove the TEL files related to the options you don't want to be displayed from the FCB directory ...*<release dir>*\BIN.
2. Remove all files from the C:\PC_LIB (or PC\$LIB) directory. Normally, no files should be present here.
3. Start the Setup Add-Ons from the application icon and select “Compile only”.

Setup Add-Ons generates automatically the following files: function.ini, target.tel, target.001 and targconf.lis. The latter specifies all the actual TED files installed. Target.tel represents the binary counterpart of the function.ini. Target.001 is only a copy of the preceding target.tel version.

3.6 Create Library Program

Enter the Application Builder and create new node by defining name, node type, node and network number as well as selecting the options from the “Available Options” window.

In order to create a library program you must select the option containing the interface elements. The option with User Elements is than not needed.

When you create an application program with User Elements, you select only the option containing these elements.

If you create a library and application programs in the same node, than you select both options.

The library program consists of as many control modules (CONTRM) as number of control solutions to be implemented as User Elements. It doesn't matter which execution time you select for the control modules. The first and last element in a CONTRM must be an interface element.

When you complete the library program, you generate source code and convert it.

Such produced library source code (*.AA) together with the related PED library (*.PED) is ready for installation in the controller.

Chapter 4 User Elements in the Controller

4.1 Introduction

When you have created the PED library and the library source code by using FCB, the installation into the controller may take place. For this purpose you need the AdvaBuild On-line Builder connected to the controller. The installation process consists of two operations: download and pre-processing.

Down-load of the library source code is done in the same way as for an ordinary AMPL source code. Down-load of PED library is done by transferring the PED library file to a temporary storage data area in the controller. The area is reserved when you create an AAB_FILE element.

Pre-processing is an internal transformation of down-loaded data to executable PC elements. The installation procedure is explained in detail in [Chapter 5, Installation and Dump of User Elements in the Controller](#). In this chapter we pay attention to the data base element AAB_FILE.

4.2 DB Element AAB_FILE

The DB element AAB_FILE is applied when you want to transfer data, other than AMPL source code, to a storage data area in the controller. The size of this area and the maximum number of AAB_FILE elements you can create, is defined when you dimension the controller's data base. You can delete or overwrite the content of an AAB_FILE. The stored data in an AAB_FILE is invisible and can not be displayed in a graphical or text representation. The stored data is cleared when you apply the On-line Builder's command RECONFIG. [Figure 4-1](#) shows the terminals in the data base necessary to be considered.

NOTE

In this manual, we show you how to use terminals of the AAB_FILE element in the context of User Elements. A full description of the terminals of the AAB_FILE is found in the DB Elements Reference Manual.

AAB FILE		(323.1)
— 1 —	NAME	ERROR — 14 —
— 3 —	ALL_SIZE	LOAD_AAB — 7 —
— 4 —	VERSION	LOAD_MP — 8 —
— 5 —	REVISION	LOCKED — 9 —
—10 —	WR_PROT	F_SIZE — 15 —
		SOURCE — 16 —
		F_DATE — 17 —
		F_TIME — 18 —
		DATA_PTR — 19 —

Figure 4-1. Data Base Element AAB_FILE

Table 4-1. Data Base Element AAB_FILE Terminals

Terminal No Name	Entered By	Default Value	Data Type	Description
1 NAME	User System	FPS_SDES<n> ⁽¹⁾ FPS_CLIB<m> ⁽¹⁾	A20	Unique name. Max. 20 characters. AAB_FILE elements created by the system ⁽¹⁾
3 ALL_SIZE	User System	64	I*2	Range 1 - 64 kB. Size of allocated storage data area. Max. size of a file that can be down-loaded is 64 kB
4 VERSION	User	0	I*2	Range 1 - 7. Command number you enter to identify an operation related to installation and maintenance: 1 - Pre-process PED data (See Chapter 5.3.1) 2 - Pre-process library code (See Chapter 5.3.2) 3 - Perform total library installation (See Chapter 5.8) 4 - Display spare area factor (See Chapter 6.5) 5 - Change spare area factor (See Chapter 6.5) 6 - Vacant. Reserved for future use. 7 - Force PED load (See Chapter 6.4) ONLY FPS_WORK, ELSE NOT USED ⁽²⁾

Table 4-1. Data Base Element AAB_FILE Terminals

Terminal No Name	Entered By	Default Value	Data Type	Description
5 REVISION	User	0	I*2	Range 1 - 100. Identifier for library program and PED library that are target for pre-processing. Also used to insert spare factor value (see Section 6.5) ONLY FPS_WORK, ELSE NOT USED
7 LOAD_AAB		0	B	Not used
8 LOAD_MP		0	B	Not used
9 LOCKED		0	B	Not used
10 WR_PROT	User	0	B	1 = start the operation assigned on terminal VERSION. When the operation is completed successfully , the value of this terminal is reset automatically. ONLY FPS_WORK, ELSE NOT USED
14 ERROR	System	0	I*2	Error code indication. First error detected during the most recent pre-processing. ONLY FPS_WORK, ELSE NOT USED
15 F_SIZE	System	0	I*4	Actual size in bytes of last down-loaded file. Always less or equal than ALL_SIZE
16 SOURCE	System	-	A12	Indicates the source for pre-processed data. The designation created by the system is: For PED library: "SD<identifier>" ⁽¹⁾ For library PC program: "PC<program no.>" ⁽¹⁾
17 F_DATE	System	-	I*4	Indicates date of AAB_FILE element creation
18 F_TIME	System	-	I*4	Indicates time of AAB_FILE element creation
19 DATA_PTR	System	324,1	I*4	Data pointer to the first LR in the storage data area segment where the content of this AAB_FILE element resides

(1) See [Section 4.4, Statement DEscription and Code LIBrary](#)

(2) On FPS_WORK see [Section 4.3](#)

4.3 Applying FPS_WORK Element

The first AAB_FILE you have to create before you start the installation must be named "FPS_WORK". This is a reserved name applied when working with User Elements. This DB element is used for several purposes:

- It reserves permanently a working data area in the controller where a down-loaded PED library data resides prior to pre-processing. The content of this data area is overwritten each time a new down-load is requested. Therefor a designation temporary storage area is applied.
- You can request an operation related to installation or maintenance by entering a command number on terminal VERSION. The requested operation will be started as soon as you enter value 1 on the terminal WR_PROT.
- Error codes and other information regarding the installation, are preserved and displayed on the output terminals.

Terminals marked with "**ONLY FPS_WORK, ELSE NOT USED**" in the [Table 4-1](#) are used only for this AAB_FILE element. These terminals have no function for other AAB_FILE elements created by you or automatically.

You can delete and re-create the FPS_WORK element with full or diminished size, providing the name rule is preserved. This is necessary to avoid conflict if the AAB_FILE elements are utilized for other purposes in the controller.

The presence of FPS_WORK element in the controller is not needed after a successful installation of all libraries. By deleting it, you actually release the reserved storage data area.

4.4 Statement DEScription and Code LIBrary

Internal system designation for PED library's data in the controller is Statement DEScription library. Abbreviated: **SDES**. Internal system designation for library source code's data in the controller is Code LIBrary. Abbreviated: **CLIB**

SDES

During the pre-processing of a PED library, an AAB_FILE element named FPS_SDES<n> is automatically created. This name is the system default and you can modify it. Value of the parameter <n> depends on installation order as explained below.

The content of a SDES element is identified in form SD<identifier> and displayed on terminal SOURCE. Parameter <identifier> is picked up from FPS_WORK's terminal REVISION at pre-processing start. You can rename the created SDES to project or product specific designation.

The real internal size (bytes) of loaded data is visible at terminal F_SIZE. The terminals F_DATE and F_TIME indicates date and time of the most recent successful pre-processing.

CLIB

During the pre-processing of a library source code, an AAB_FILE element named FPS_CLIB<*m*> is automatically created. This name is the system default and you can modify it. Value of the parameter <*m*> depends on installation order as explained below.

The content of a CLIB element is identified in form PC< *program no.*> and displayed on terminal SOURCE. Parameter < *program no.*> is picked up from FPS_WORK's terminal REVISION at pre-processing start. You can rename the created CLIB to project or product specific designation.

The real internal size (bytes) of loaded data is visible at terminal F_SIZE. The terminals F_DATE and F_TIME indicates date and time of the most recent successful pre-processing.

Installation Order

If there are more than one library to be installed, you can decide the installation order.

The following is recommended:

- Installation of a PED library is followed by installation of the related library source code.
or
- Installation of all PED libraries is followed by installation of all library source codes in the same order as the related PED libraries.

NOTE

A SDES must exist in the controller before the related CLIB is created.

Chapter 5 Installation and Dump of User Elements in the Controller

5.1 Getting Started

After a controller cold init./RECONFIG, the data base must be dimensioned. The User Element handling requires that the data area for AAB_FILE elements is properly dimensioned **in addition** to the rest of your application. To do so:

- Start On-line Builder session by selecting Accuray Board handling from the expanded main menu (More...). Select "REMOTE - Direct" if your engineering station is physically connected to the controller.
- During DIMDB open the S4 image - DATA TABLES & TREND DATA.
- Modify the terminal -L323- Number of FILE ELEMENTS - to wanted number of AAB_FILE elements in the DB. Number of elements = 2*number of libraries + 1. A minimum of three elements is required to obtain a working system.
- Modify the terminal -L324- Size of FILE DATA (kb) - to the wanted number of kilobyte internal DB area allocated to the AAB_FILE elements. For calculation of memory consumption and required DB area for AAB_FILES see [Chapter 7, Performance](#). The following rule of thumb is recommended:
 $2 * \text{largest PED file} + 0,25 * \text{SUM of all library source code files (which is AA files)}$.
A minimum of three kilobyte must be specified.
- Remember to dimension PC, DIMPC.

NOTE

We recommend you to over-dimension the size of FILE DATA until experience with total memory consumption is gained.

The remaining free space in the system must be at least 64 kB after dimensioning of DB and PC area is performed.

5.2 Down-Load

Down-load of a library source code is done by applying On-line Builder's command TRPCS, that is, in the same way as for any ordinary AMPL source code. Before you do that you must down-load and pre-process the related PED library. In this section we take a look at how to down-load a PED library.

PED Library File in the SRCE Directory

The PED library file that you create by a text editor in Windows environment has an extension of three letters. To be recognized by the On-line Builder it must have extension with maximum two letters. If you decide to keep this file together with the related library source code (*.AA) in SRCE directory, then change the extension PED to PE and copy the file to directory SRCE of the controller concerned. Although another extension can be applied, the PE is recommended.

Creation of FPS_WORK Element

Enter the system level and create the FPS_WORK element:

```
* CRDB AAB_FILE
> M
NAME: FPS_WORK
ALL_SIZE: <all_size> ! followed by CR          Dots indicates creation of
                                                data area segment.

FPS_WORK IS SUCCESFULLY CREATED
```

Allocated size <all_size> must be equal or higher than the largest PED library file intended for down-load. We recommend you to increase this value for possible future enlargements.

Down-load a PED Library

The next step is down-load of a PED library. This is done by applying On-line Builder's command LOADFILE. The command syntax rule is:

```
*LOADFILE FPS_WORK,<directory>:<filename.extension>
```

Alternatively:

```
*LOADFILE followed by CR
Name of DB element:   FPS_WORK followed by CR
File to load:         <directory>:<filename.extension> followed by CR
H'<n> bytes successfully loaded
```

The DB element name is always FPS_WORK. Default <directory> is SRCE but you can enter any directory on the available paths in Windows environment. If the entered <directory> is SRCE or USER, the file extension may not have more than two characters.

Data stored in FPS_WORK can now be pre-processed. The content of FPS_WORK is overwritten each time you down-load a new PED library. The question OVERWRITE ? is displayed when the second or any further down-load is requested. Answer with "Y" to continue the installation.

5.3 Pre-processing

5.3.1 Pre-processing PED Library

Pre-processing of down-loaded data is started by identifying the operation, and setting FPS_WORK's terminal WR_PROT to true:

- Set the controller in P2/ECONFIG mode.
- Give the following commands in the AdvaBuild On-line Builder:

```
*MDB FPS_WORK
>M
VERSION: 1
REVISION: <identifier>           Preferably libr. program No.
WR_PROT: 1 ! followed by CR
>END
pre-processing START>
pre-processing STOP
```

NOTE

Never use 0 (zero) as identifier! We strongly recommend you to use the library program number.

If any error occurs, a system message is displayed before the “pre-processing STOP” line. If there are no additional messages, this PED library is successfully pre-processed and a statement description (SDES) is created. Quick reference guide to system messages that are likely to appear during the pre-processing of statement description is given in [Table 5-1](#). For details on error reporting see [Chapter 9, Error Codes and Messages](#).

Table 5-1. Quick Reference Guide to System Messages

Trace Code	Data1	Reason/Recommended Action
03052066	Error	Number of errors detected during pre-processing of PED library.
03052068	---	Two User Elements defined with equal names. Action: Change one of the names.
0305206A	---	User Elements name equal to name of a standard PC element. Action: Change the name of the User Element.
0305206E	---	Interface element name equal to standard PC element. Action: Change the name of the interface element.
0305206C	---	The User Element exists in another library. Action: Change the name of the User Element.
03052070	<n>04	Attempt to Pre-process a User Element with illegal changes (<n> = User Element order No. in PED library). Only change of terminal name allowed.

Table 5-1. Quick Reference Guide to System Messages (Continued)

Trace Code	Data1	Reason/Recommended Action
0305207C	30002	Not possible to add another AAB_FILE. Action: Re-dimension number of AAB_FILE elements.
0305207A	---	Impossible to create a big enough AAB_FILE element. End of dimensioned data area. Action: Re-dimension the space for the AAB_FILE elements.
03052078	---	The data area for temporary storage of PED libraries is damaged by an illegal operation. Action: Delete and re-create a new FPS_WORK.

During the pre-processing, one interface element is created for each User Element. Both interface and User elements are now available in the controller along with the standard elements.

Installed User Elements can be inserted in an application program designed in build mode of the On-line Builder. Such created application program is, however, not executable until down-load and pre-processing of the library source code is done. Down-load is done by using the On-line Builder:

```
* SLLEV PC
# TRPCS SRCE: <library code>
TRANSLATION OKAY
```

5.3.2 Pre-processing Library Code

Pre-processing of a down-loaded library program is started by identifying the operation, and setting FPS_WORK's terminal WR_PROT to true.

- The controller is set in P2/ECONFIG mode.
- The following commands are given in the On-line Builder

```
# MDB FPS_WORK
>M
VERSION: 2
REVISION: <program no.>
WR_PROT: 1 ! followed by CR
>END
pre-processing START
pre-processing STOP
```

If any error occurs, a system message is displayed before the “pre-processing STOP” line. If there are no additional messages, this library source code is successfully pre-processed and one Code Library (CLIB) is created. A quick reference guide to system messages that are likely to occur during the pre-processing of code library is given in [Table 5-2](#). For details on error reporting, see [Chapter 9, Error Codes and Messages](#).

Table 5-2. Quick Reference Guide to System Messages

Trace Code	Data 1	Reason/Recommended Action
03053028	84	DIMPC not done
03053028	85	PC program number not dimensioned.
03053028	86	PC program not found in PC system table.
03053025	<n>59	Illegal structure PC element found in the library program (<n> = consecutive no. of User Element's CONTRM). Action: Remove illegal PC element(s).
03053025	56	CONTRM without PC elements found in the library program. Action: Redesign.
0305300A	<n>74	No. of terminals of inserted interface input element, do not match input terminals of the related User Element (<n> = User Element's order no. in the PED library). Action: Check total definition of the User Element and pre-process.
0305300A	<n>75	The same problem as above but for output terminals (<n> = User Element's order no. in the PED library). Action: Check the total definition of the User Element and pre-process.
0305300C	<n>68	Interface element is missing or not the first/last PC element in a CONTRM (<n> =consecutive No. of User Element's CONTRM). Action: Redesign.
03053006	<n>76	Illegal DB connection attempt (<n> = consecutive no. of User Element's CONTRM) Action: Remove DB connection(s).
0305300E	---	Attempt to pre-process a library code containing two User Elements with equal names. Action: Remove duplicate.
03053018	30002	Not possible to add another AAB_FILE. Action: Re-dimension number of AAB_FILE elements.
03053016	30003	Not possible to create big enough AAB_FILE. End of DB data area Action: Re-dimension space for AAB_FILE elements.
03053014	30004	Not possible to reinstall existing AAB_FILE Action:Re-dimension AAB_FILE element.

After a successful pre-processing, the User Elements installed in the controller are executable. An application PC program with User Elements may now be down-loaded and executed at any time.

5.4 User Elements in an Application Program

The installation process described in [Section 5.2](#) and [Section 5.3](#) is to be repeated for the next library. Be aware of that the content of FPS_WORK element is preserved after an installation. The question OVERWRITE ? is displayed when second or any further down-load of PED libraries is requested. Answer with "Y" to continue the installation.

The down-loaded library program can be removed from the controller after a successful installation. An application program with the same number as the removed library program can be entered or down-loaded, but this is not recommended. You should reserve a sequence of numbers dedicated for library programs, for instance PC90 to PC99.

You can down-load only application programs with User Elements present in the controller.

It is important to note that an application program with User Elements can be created in build mode, providing the PED library concerned is pre-processed. Such created program is not executable before the related library program is down-loaded and pre-processed.

NOTE

Attempt to run an application program with missing or insufficient SDES or CLIB, will set the controller in configuration mode and stall it intentionally.

A halt code is displayed on the CPU card. If this happens, a corrective action must be taken before new start attempt is made. More detailed information about errors can be found in [Section 9.2, Error Codes](#).

5.5 Installation Example

We will now look at an example of installation of a library with your control solutions. PC element description for this library is stored in MYLIB.PED and library source code in MYPC9101.AA in the AdvBuild for Windows environment. Do as follows:

1. Make the PED library file available in the SRCE directory concerned.
 - Rename the MYLIB.PED to MYLIB.PE.
 - Copy MYLIB.PE to the ...\SRCE directory for the actual controller.
2. Start the On-line Builder towards the wanted controller
3. Dimension the data base
 - DIMDB
 - >S4
 - >M
 - L323: 3
 - L324: 80 ! followed by CR
 - >DIM
 - DB SUCCESFULLY DIMENSIONED

Dimension also your application DB.

4. Create FPS_WORK

```
– *CRDB AAB_FILE
  >M
  NAME: FPS_WORK
  ALL_SIZE: 20 ! followed by CR
  >END
```

Dots indicates creation of storage data area segment.

5. Load the PED library file into the controller.

```
– *LOADFILE
  Name of DB element: FPS_WORK followed by CR
  File to load: MYLIB.PE followed by CR
  <n> bytes successfully loaded
  Load AAB_FILE completed successfully
```

6. Pre-process the PED library.

```
– *MDB FPS_WORK
  >M
  VERSION: 1
  REVISION: 91
  WR_PROT: 1 ! followed by CR
  >END
  pre-processing START
  pre-processing STOP
```

FPS_SDES2 is created.

7. Dimension the memory size for PC programs

```
– * SLLEV PC
  # DIMPC
  >M
  Number of PC programs: 99
  Size of USER disk : 50 ! followed by CR
  >DIM
  PC SUCCESSFULLY DIMENSIONED
```

Take your application programs into account too.

8. Down-Load the library source code

```
– # TRPCS SRCE: MYPC9101
  TRANSLATION OKAY
```

9. Pre-process the library code

```
– *MDB FPS_WORK
  >M
  VERSION: 2
  REVISION: 91
  WR_PROT: 1 ! followed by CR
  >END
  pre-processing START
  pre-processing STOP
```

FPS_CLIB3 is created

5.6 Dump and Load of Total Data Base and Library Programs

The installation procedure described in the previous sections does not need to be performed for each single cocontroller in a network configuration. A dump of a total data base or application, created in one controller may be transported and loaded into another. The condition is that the recipient controller is of the same type and with identical standard PC libraries and software options (internal software composition).

A dump of total DB and application is performed in the ordinary way, by applying On-line Builder's commands DUTDB and DUAP.

The selected controller for production of libraries (see [Section 6.3, Changing Libraries in a Dedicated Controller](#)), that is, creation of elements SDES and CLIB, shall be dimensioned only for AAB_FILE elements. The library programs can be removed after the pre-processing. They do not need to be present in your total application dump (DUAP).

The FPS_WORK element may also be removed.

During the down-load in another controller, started by the commands LOTDB or LOAP the following messages, related to AAB_FILE elements, are displayed:

FPS_WORK successfully created	Appears only if the FPS_WORK has been included in the dump
SDES successfully installed	Appears for each installed library
CLIB successfully installed	Appears for each installed library

When you apply LOTDB

On the question:

REDIMENSION ?
you answer:
Y
in order to properly dimension the memory area for your DB application.

When you apply LOAP

After a successful installation in the destination controller you must create a total DB dump by applying DUTDB command. This, in order to re-dimension data base area for your application.

If the library programs are included and you want to keep them in the destination controller, then you must make a dump of program tables by applying DUTPT command.

5.7 Dump and Load of Single Library

Dump of single library suppose dump of a SDES and related CLIB. Ones created by a pre-processing, both AAB_FILE elements can be dumped to files named pre-processed library files. These files can then be transferred to another controller and loaded to AAB_FILE elements created for this purpose.

The above is possible only when the destination controller is of the same type and with identical internal software composition as the controller used for creation of the SDES and CLIB.

Dump of a single library is done by On-line Builder's command DUMPFIL. The command syntax rule is as follows:

* DUMPFIL <directory>:<filename.extension>,<AAB_FILE name>

Alternatively:

* DUMPFIL followed by CR

File ? : <directory>:<filename.extension> followed by CR

Name of DB element ? : <AAB_FILE name> followed by CR

H'<n> bytes successfully loaded

The <AAB_FILE name> is name of the SDES and related CLIB, respectively. Default <directory> is SRCE but you can enter any directory on the available paths in Windows environment. The file name must be specified with an extension that may have max. 3 characters, but be aware that the directories SRCE and USER accept only two characters. The extension should, by convenience, be SD for statement description and CL for code library.

Load Pre-Processed Library Files

In order to load a single library, pre-processed library files must be present and available to the destination controller. You may copy them to the actual SRCE directory. Assuming the data base is properly dimensioned, you shall now create the FPS_WORK element in the destination controller:

* CRDB AAB_FILE

> M

NAME: FPS_WORK

ALL_SIZE: <all_size> ! followed by CR

FPS_WORK IS SUCCESFULLY CREATED

Dots indicates creation of
Storage data area segment.

The FPS_WORK element will only be used to activate a total library installation as explained in [Section 5.8, Total Library Installation](#).

Allocated size <all_size> does not need to be more then 1, assuming no down-load of PED libraries is planned. Create two AAB_FILE elements for SDES and CLIB respectively. You should enter names that identify their content and interdependence. The allocation size (terminal: all_size) shall be the same as the size of the pre-processed library file concerned (rounded down to the nearest integer value represented in kB).

Now the LOADFILE can be applied. The command syntax is:

```
* LOADFILE <AAB_FILE name>,<directory>:<filename.extension>
```

Alternatively:

```
* LOADFILE followed by CR
```

```
Name of DB element ? : <AAB_FILE name>
```

```
File to load ? : <directory>:<filename.extension> followed by CR
```

```
H'<n> bytes successfully loaded
```

Default <directory> is SRCE but your pre-processed library files may be located in any directory on the available paths in Windows environment.

Create two new AAB_FILE elements for the next library and apply the LOADFILE command again. When all your libraries are loaded in the controller, a total library installation shall be performed as explained in the next section.

5.8 Total Library Installation

Total library installation is an internal update of all loaded SDES and CLIB elements and may be performed at any time provided the controller is in P2 mode.

After successfully loading of all SDES and CLIB as described in [Section 5.7](#), a total library installation is to be carried out as follows:

```
*MDB FPS_WORK
```

```
>M
```

```
VERSION: 3
```

```
REVISION: <some integer>
```

Not applicable for this operation.

```
WR_PROT: 1 ! followed by CR
```

```
>END
```

```
SDES INSTALLED
```

Two messages for each library installed.

```
CLIB INSTALLED
```

The installed User Elements are now available in the destination controller and your application programs can be down-loaded at any time.

Chapter 6 Change Handling and Maintenance

6.1 Change Handling

Be aware of the following when you modify an existing library or a single User Element:

- Any changes must take place with the controller in P2/Configuration mode.
- All application programs present in the controller are forced automatically to BUILD MODE when the pre-processing is activated.
- A single library created in one controller may be installed in an other, providing compatible software modules are present in the destination controller.
- A single library is built up of two AAB_FILE elements, containing a PED library and a library source code respectively.
- A User Element is a member of a library that may contain many User Elements. Change of a single User Element may affect the whole library. The running application programs with User Elements concerned may also be affected.

NOTE

Before any modification takes place, make sure that source code dump of all running application programs is available.

Add a New User Element to an Existing Library

When a User Element is added to an existing library, a re-installation of the whole library is required. All application programs with User Elements must then be down-loaded again. The only exception is when you add a new User Element at the end of the last installed library in a controller. Re-installation and down-load of all application programs is often undesirable because of short dead-lines or other reasons. As a temporary solution, we recommend you to make a new library containing only the new User Element.

Remove an Existing User Element from a Library

You may remove a User Element from a library, providing that you remove it from the application programs. Otherwise, the controller can not be started in operational mode.

Both the User Element's description in the PED library, and its control module in the library program must be removed. A new installation of the whole library is needed. All application programs must be down-loaded again. The only exception is when the removed User Element was placed at the end of the last installed library in a controller. In this case only the affected application programs must be down-loaded again.

Add a New Library

You can add new libraries in a controller, providing that you have dimensioned a sufficient number of AAB_FILE elements and data area size. We recommend you to dimension spare area for at least one more library than strictly required. After a new library is installed by pre-processing, the library program can be removed. However, it may be present in a controller side by side with the application programs.

Remove an Existing Library

If a whole library is removed (deleted), all application programs using the User Elements from that library must be deleted in advance. The other application programs are not affected.

To remove a library from a controller, both AAB_FILES (SDES and related CLIB) shall be deleted.

6.2 Performing Changes when the Library Programs are Available

You can change an existing User Elements, assuming that the library program concerned is present in the controller and the related PED library file is available.

Changing a User Element's Description

Only a User Element's terminal names may be changed without need for down-load of the affected application programs. After re-installation of the PED library, the modified terminal names become visible in the application programs. All other modifications such as adding new terminals, change of data types, deleting terminals etc. requires change and new down-load of the affected application programs.

PED library with minor changes, other than terminal names, can be installed by "force". For details see [Section 6.4](#).

Changing a User Control Solution

The user's control solution may be changed without down-loading neither the library program nor the application programs.

NOTE

Modification performed in one controller may complicate the traceability in case the same library is used in other controllers within a project. Every change of a control solution must therefor result in a new revision, indicated by the name of a revision terminal, for example "REV_<rev.id.>".

Follow the procedure below if you have to perform changes in the control solution of a User Element applied in several application programs running in a controller:

1. Set the controller in P2/Configuration mode.
2. Enter the library program with the On-line Builder and perform the modification.
3. Enter the relevant PED library file, and change the revision terminal.
4. Down-load and pre-process the PED library.
5. Pre-process the library code.
6. Set the controller in P1/Operational mode and disable the build mode.

If you now enter an application program with the modified User Element, the new revision number (terminal name) is displayed.

6.3 Changing Libraries in a Dedicated Controller

A large control network configuration with many nodes is usually composed of different controller types with various standard PC libraries and software options. We recommend you to identify groups of controllers that are equipped with the same software modules containing standard PC element libraries and options. Any member of the group represents the whole group, regarding the internal software composition. One of the controllers from a group can be dedicated for production and maintenance of User Elements. The benefits are:

- Creation, modification and maintenance of libraries are centralized and full traceability achieved. Revision inconsistency and faults caused when attempting to install the wrong versions of User Elements are minimized.
- Pre-processing and down-loading of PED libraries and source codes in other controllers are not necessary.
- New created or modified libraries are dumped to pre-processed files and distributed to all users concerned, even to remote locations by transfer through electronic mail system.
- Protected control solutions can be delivered in form of pre-processed files that are not possible to read. Source code is then maintained by the company of solution's origin.

Any controller can actually be applied as dedicated, providing the controller type and its internal software composition is identical with the recipient controller.

Working Flow when Modifying in a Dedicated Controller

Assume that you perform modifications in one or several User Elements that are constituents of the same library. Such modified library shall then be loaded into another controller, applying an old version of the User Elements concerned. The following tasks are to be performed:

1. Delete the affected control modules in the library program concerned (always present in a dedicated controller)
2. Delete the affected AAB_FILE elements, that is, SDES and CLIB.
3. Perform the required modifications in the PED file by a text editor.
4. Modify the affected control solutions in FCB and create a new source code for the control modules concerned.
5. Down-load and pre-process the modified PED library.
6. Down-load the control modules concerned and pre-process the library code.
7. Create pre-processed library files (consisting new created SDES and CLIB).
8. Transfer the pre-processed library files to SRCE directory of the destination controller.
9. Delete all applications in the destination controller that utilize the User Elements concerned.
10. Create FPS_WORK and two other AAB_FILE elements in the destination controller.
11. Load the pre-processed library files into the destination controller and perform total library installation.
12. Modify and down-load the application programs.

6.4 Force PED Load

Forced load of a PED library is primarily applied when the libraries are produced and maintained in a dedicated controller.

NOTE

You must delete both the control modules in the library program, and the application programs affected by a change, before you activate the force load command. Otherwise, the whole library program and the affected application programs must be deleted because of interface elements' inconsistency.

Assume you want to perform minor modification in a User Element's description (not only terminal name) and control solution. The library program concerned is present in the controller. As the controller is dedicated for production of libraries, it does not contain any application programs.

Normally, you should perform the six tasks of the working flow described in [Section 6.2](#).

In order to save time and reuse the existing PED library (SDES), already present in the controller, it is desirable to simplify the installation process when minor changes are made.

For this purpose, a command for forced load of a PED library is available. It is called forced as the existing SDES (storage area in the controller) is “overwritten”. Normally, you should delete this and create a new by pre-processing. Follow the procedure below:

1. Set the controller in P2/Configuration mode
2. Enter the library program with the On-line Builder and delete the control module concerned.
3. Enter the relevant PED library file by a text editor and perform the changes. Do not forget the REVision terminal.
4. Down-load the new PED library and perform forced installation as follows:

Enter the system level with the On-line Builder

*MDB FPS_WORK

>M

VERSION: 7

REVISION: <identifier>

Normally, library program No.

WR_PROT: 1 ! followed by CR

>END

pre-processing START

WARNING ! Dangerous operation

pre-processing STOP

5. Modify the affected control solution in FCB and create a new source code for the control modules concerned.
6. Down-load the control modules and pre-process the whole library

The warning displayed during the pre-processing is only a reminder.

6.5 Administrate Spare Factor

When a library code is pre-processed, the required memory size of the local data (LD) area is calculated for each User Element. Suppose you want to modify a User Element already present in the application programs in a running controller. If the modification results in an increased number of PC elements, the required LD area size will also increase. After new pre-processing, such enlarged User Element will allocate more memory each time it is inserted in an existing application program. When the LD area for an application program (limited to 32 kB) is full, the allocation of memory will take place out of the storage area reserved for the LD variables. Source code dump of such application will not be possible to down-load in an other controller because of exceeded LD area size. The application program must then be split up. To prevent or minimize this type of memory allocation it is possible to specify a factor by which calculation of required LD area size will be initially multiplied with. This spare factor is by default 0. When higher than 0, the required LD area is extended proportionally with the factor's size.

This extension (spare area) is calculated for each appearance of a User Element in an application program. However, to make this approach effective, the spare factor must be defined prior to pre-processing of a PED library and before down-load of the application programs utilizing the User Elements.

Alteration of spare factor value shall be considered when major changes of an existing control solution are predicted. Especially, when a modified and enlarged User Element appears many times in an existing application. A spare factor value in range 5 to 10 % is then recommended. When minor changes are expected, no additional spare area is needed.

Displaying Factor Value

To read the current factor value, switch the controller to P2/ECONFIG mode and start by identifying the operation, and setting FPS_WORK's property WR_PROT to true:

```
*MDB FPS_WORK
>M
VERSION: 5
WR_PROT: 1 ! followed by CR
>END                                     No system feedback

*MDB FPS_WORK
REVISION: <current value>
```

Set Spare Factor

To set a new factor value, switch the controller to P2/ECONFIG mode and start by identifying the operation, and setting FPS_WORK's property WR_PROT to true:

```
*MDB FPS_WORK
>M
VERSION: 4
REVISION: <% spare>                     Enter value in range 0 - 100 %
WR_PROT: 1 ! followed by CR
>END                                     No system feedback.
```

Chapter 7 Performance

7.1 Load Calculation

Load calculation for User Elements as well as interface elements should be performed and documented. Calculation of execution time in this section assume that all User Element's terminals are connected to other elements executed by the same interpreter. When a User Element is inserted in an application program, additional time is to be calculated for each connection via read-in and read-out elements, as well as connections to data base elements DOS and AOS. Size of this addition is explained in relevant PC element manuals.

Execution time (μS)

The Execution time for an interface input element is constant;

4 μS for Advant Controller 410, and 1 μS for Advant Controller 450

Execution time for an interface output element is constant;

6 μS for Advant Controller 410, and 2 μS for Advant Controller 450

User Element's execution time is calculated in accordance with the equation:

$$K + E + \sum (Td * Mf)$$

K - Constant Depends on which controller:
14 for Advant Controller 410
4 for Advant Controller 450.

E - Total execution time for all PC elements (beside the interface elements) in the User Element's CONTRM. The structure element CONTRM itself is not taken into account.

Td - Terminal count of same data type.

Mf - Multiplying factor as indicated in [Table 7-1](#)

Table 7-1. Mf for Load Calculation

Data Type:	B, I, IL, R, TR	T	A, G
Multiplying factor			
AC 410	4	6	14
AC 450	1	2	4

7.2 Memory Consumption

Space requirements for User and interface elements, as well as memory requirement for the library code, should also be performed and documented. Calculation of memory consumption in this section assume that all User Element's input terminals are connected to outputs of other elements, executed by the same interpreter. When a User Element is inserted in an application program, additional space is to be calculated. This addition takes into account connections to operational parameters, constants, data base elements and outputs of a different priority. Size of this addition is explained in relevant PC element manuals.

PC Statement (bytes)

Memory consumption for an interface element is $8 + 2 * T_c$. T_c is terminal count

Memory consumption for a User Element inserted in an application program is equal to total memory consumption for all PC elements in the User Element's control module inclusive the interface elements. The structure element CONTRM itself is not taken into account.

Local Data Area (bytes)

Memory consumption for an interface input element is calculated in accordance with the equation:

$$\sum (T_d * M_f)$$

T_d - Terminal count of same data type
 M_f - Multiplying factor as indicated in [Table 7-2](#).

Memory consumption for an interface output element is always 0 (zero)

Memory consumption for a User Element is calculated in accordance with the equation:

$$18 + E_i + \sum (T_d * M_f)$$

E_i - Total memory consumption for all PC elements in the User Element's CONTRM including the interface input element. The structure element CONTRM itself is not taken into account.
 T_d - Terminal count of same data type. Only input terminals within the User Element's CONTRM connected to constant.
 M_f - Multiplying factor as indicated in [Table 7-2](#).

Table 7-2. M_f for LD Memory Consumption

Data Type:	B, I	IL, R, T, TR	A, G
Multiplying factor	2	4	$2 + N^{(1)}$

(1) N is a number of bytes in the data type definition.
For instance $N = 12$ when the data type is A*12.

We recommend you to use multiple constants where applicable in order to reduce the memory requirements for local data area.

7.3 Library Size

The size of a library code (CLIB) is the total amount of memory requirements (PC statement only) for all User Element's control modules in the library program. This size is to be taken into account when dimensioning the controller for data base elements (terminal L324 - size of FILE DATA, in the On-line Builder's dimensioning window).

7.4 Calculation Example

Assume the User Element VALVE is the only element in a PED library and the corresponding library program, required to be installed in an Advant Controller 410.

The User Element VALVE has 11 input and 7 output terminals distributed as listed below.

Table 7-3. Distribution of Input and Output Terminals

Input Terminals		Output Terminals	
Data Type	Count	Data Type	Count
B	7	B	4
R	1	I	1
TR	1	G 10 I	1
T	1	A*12	1
A*32	1		

Number of standard PC elements included in the control module is 25. Their total execution time is 457 μ S. Their memory consumption is 385 bytes for PC statements and 117 bytes for local data area. There are four terminals in the control module connected to constants: one of type boolean, two of type real and one of type A * 12.

Calculation of execution time, memory consumption and library size will be as follows.

Execution Time

Interface input element: 4 μ S

Interface output element: 6 μ S

User Element: $14 + 4 + 6 + 457 + (14 * 4 + 1 * 6 + 3 * 14) = 585 \mu$ S

Memory Consumption, PC Statement

Interface input element: $8 + 2 * 11 = 30$ bytes

Interface output element: $8 + 2 * 7 = 22$ bytes

User Element: $30 + 385 + 22 = 437$ bytes

Memory Consumption, Local Data Area

Interface input element: $7*2 + 3*4 + 1*34 = 60$ bytes

Interface output element: 0

User Element: $18 + 60 + 117 + 1*2 + 2*4 + 1*14 = 219$ bytes

Library Size

As the number of User Elements in the library program is only one, the memory consumption for the library is 437 bytes.

Chapter 8 Test and Documentation

8.1 Test

A User Element inserted in an application program can be black-box tested by manipulation of input terminals and observation of output terminals. We recommend you to perform detailed test prior to installation and translation of application programs, especially when the User Element's control solution is complex.

The internal states of the User Element's control solution can not be observed. The library program is totally inactive in the system after an installation and has the main purpose of serving as input source for pre-processing the library code.

Although the library program can be removed after the installation, we recommend you to keep it during the functional test of application programs.

If internal states in the User Element must be observed, it can be done by temporary replacement of the input and output interface elements with MOVE elements. The inputs are then manipulated directly in the library program. The interface elements must be restored manually after fulfilled test. A library program with MOVE instead of interface elements can be tested in a controller without the User Element software module.

When a minor modification of the control solution is assumed, it is not necessary to apply MOVE elements. Changes are made directly in the library program. After the pre-processing the changes are available in all application programs containing the User Elements concerned.

8.2 Documentation

The documentation should be made for the whole library as an integral software package or for a single User Element, as appropriate. The documentation should include:

- PC element description and graphical image of the User Element(s)
- PC diagram printout of the User Element control design
- Functional description for each single element
- Terminal description for all User Element's terminals
- Load and memory consumption
- Minimum system software requirements for the whole library or each single element
- Dependency on other User Elements
- Test procedure for a single element and/or group of elements interconnected in a control solution.

Technical design specification should also be included for elements with a complex algorithm.

Chapter 9 Error Codes and Messages

9.1 Message Type and Format

An example of a typical system message layout is shown below.

11	21*19:32:17	70	20	FAFA100	A	H'305301A	H'25	3781
↑	↑	↑	↑	↑	↑	↑	↑	↑
netw	node	time	err. code	task	aborted	address (trace No.)	data1	data2

The only specific message type reported by the User Element software module is 70. The task name always begins with the letters **FAF**. Each system message gives an error code in addition to the message type. A list of standard error codes reported by the User Element software module is given in [Table 9-1](#).

Table 9-1. Error Codes Standard

Code	Constant Name	Description
0	fx\$NoError	No error - (test messages only)
13	fx\$in_param	Error in input parameter
14	fx\$task_param	Task parameter error
20	fx\$ill_action	Illegal action
21	fx\$data_lost	Input/system data is lost
47	fx\$ResErr	Resource error

9.2 Error Codes

The specific error codes defined in the User Element software module appear as “data1” or “data2” in the system message layout. These codes, listed in [Table 9-2](#), are grouped in two classes:

- LE Logical errors. Normally, solved by the user.
- SE System errors. Knowledge of internal system routines is required for trouble shooting these errors.

Any system messages with error codes of class SE (System Error) shall be reported to the supplier of the User Element module.

During the installation process, different system messages may occur due to incorrect input data. Some errors may follow from others, and may seem out of context (not likely to occur), but those will vanish as the original error is corrected.

In case of multiple errors, the last message displayed indicates the number of errors detecting during the pre-processing.

NOTE

Some of the codes include an integer referring to a User Element consecutive number in a library code or a statement description. Example:

11 21*19:32:17 70 13 FAFA200 A H'3052004 310 17

Data1= 310 in this example refers to error code 10 (terminal name to long) arising out of the third User Element in a statement description (SDES).

Data2= 17 refers to a line number in a statement description, where the discovered fault resides.

To find out the exact meaning of a system message, check out the description given for each trace code in [Appendix A, System Messages](#).

Table 9-2. Error Codes User Element Specific

Code	Class	Description / Reason
0	LE	Test or reset value.
1	LE	User Element name faulty.
2	LE	User Element size error (internal data area size).
3	LE	Internal function table size is exceeded. No more space for new User Elements.
4	LE	Illegal changes in User Element's description detected. Only change of terminal name allowed.
5	LE	Error(s) detected in PED library.
6	LE	Empty *.PE file.
7	LE	Unexpected end of comment in PED library.
8	LE	Word or symbol too long (maximum 255 characters).
9	LE	Missing terminal name.
10	LE	Terminal name too long.
11	LE	Missing quotation mark or Illegal end of *.PE file.
12	LE	Missing element name.
13	LE	Element name too long.
14	LE	Unexpected end of *.PE file.
15	LE	Keyword "terminal" found between parenthesis.
16	LE	Keyword "function parameter" found between parenthesis.
17	LE	Line end (semicolon) within an illegal terminal name.

Table 9-2. Error Codes User Element Specific (Continued)

Code	Class	Description / Reason
18	LE	Missing right parenthesis.
19	LE	Illegal data type inside a Group item.
20	LE	Missing data type for members in a Group item.
21	LE	Missing number of member in a Group item.
22	LE	Missing length of string/array.
23	LE	Illegal string/array syntax.
24	LE	Unknown or missing data type.
25	LE	Wrong end of end line.
26	LE	Call parameters in element head are not allowed.
27	LE	Illegal item(s) in pc_element line.
28	LE	Illegal item in unknown line.
29	LE	Illegal line start.
30	LE	Wrong end of terminal line.
31	LE	Terminal type expected.
32	LE	Missing semicolon at User Element's description end.
33	LE	Length of array/string out of limits.
34	LE	Number of members in a group item out of limits.
35	LE	Terminal number out of limits.
36	LE	Terminal numbers not unique.
37	LE	Terminal names not unique.
38	LE	No useful lines in *.PE file found.
39	LE	End-line with no context.
40	LE	Function parameter line not allowed.
41	LE	End-line is missing.
42	LE	Start-line is missing.
43	LE	Illegal lines found between terminal lines.
44	LE	Too many terminals.
45	LE	No terminals defined in a User Element's description.
46	LE	User Element name in end line is different from the name in pc_element line.
47	LE	Definition of several terminals in one line is not allowed.

Table 9-2. Error Codes User Element Specific (Continued)

Code	Class	Description / Reason
48	LE	Illegal item(s) between terminal number and terminal name.
49	LE	Illegal item(s) between terminal name and data type.
51	SE	Internal system error. Element pointer is outside element group.
52	SE	Internal system error. Local data table size is 0.
53	SE	Internal system error. Element pointer to undefined element.
54	LE	Control block on same level as PCPGM.
55	LE	Library program with only a PCPGM element detected.
56	LE	CONTRM without PC elements in library program.
57	LE	Library program structure error. Control block on illegal location.
58	LE	PC element designation error.
59	LE	Structure element for subordinate group of elements is missing.
60	LE	Undefined data type detected.
61	LE	Too many User Elements defined in PED library. Unable to down-load file bigger than 64 Kb.
62	SE	Internal system error. Undefined element detected.
63	SE	Internal system error. Element type not initialized.
64	LE	Illegal PC structure in a library program.
65	LE	Memory allocation failure when creating temporary storage area (FPS_WORK) for PED libraries.
66	SE	Internal system error. Creation of interface element failed.
67	LE	Too many structure elements in a library program.
68	LE	Interface element is missing.
69	SE	Internal system error. End-block element out of context.
70	SE	Internal system error. Operation on empty structure stack is illegal.
71	LE	Not allowed element in element group. Presumably illegal structure element.
72	LE	Wrong start command given to the pre-processing task.
73	LE	Call or function parameter not allowed.
74	LE	Input terminal inconsistency. Interface input element in a library program contains terminals different from the corresponding User Element's terminals.
75	LE	Output terminal inconsistency. Interface output element in a library program contains terminals different from the corresponding User Element's terminals.
76	LE	Illegal DB connection inside a User Element detected.

Table 9-2. Error Codes User Element Specific (Continued)

Code	Class	Description / Reason
77	SE	Internal system error. Translator routine general error.
78	SE	Internal system error. Translator routine general error.
79	LE	Internal system error. Recursive call of subroutines is not allowed.
80	SE	Internal system error. Terminal pointer error.
81	LE	Corrupt data in library code.
82	LE	Pointer to wrong or no existing element type. Presumably structure element.
83	SE	Not used
84	LE	DIMPC not done.
85	LE	System not dimensioned for this PC no.
86	SE	Internal system error. PC program missing in index section.
87 - 95	SE	Internal system errors. Wrong or missing pointers to internal variables or storage areas.
96	LE	No terminal area in SDES found.
97	SE	Internal system error. Wrong function type found.
30001	LE	RAM buffer allocation error
30002	LE	Insufficient space for additional AAB_FILE elements.
30003	LE	Insufficient space for additional AAB_DATA elements.
30004	LE	Insufficient space for AAB_DATA expand.
30005	LE	Function table size is exceeded. No more space for additional User Elements.
30006	LE	Temporary storage area (FPS_WORK) is corrupted by the On-line Builder LODB command.
30007	LE	Pre-processing task is dormant at command time.
30008	LE	Selected SDES identifier number is out of limits (0 - 99).
30009	LE	Selected PCPGM number is out of limits (0 - 99).
30010	LE	Selected additional spare factor (%) size is out of limits (0 - 100).
30011	LE	Selected command number does not exist.
30012	LE	The controller is in wrong mode (P1/P2) for the attempted command.
30013	LE	Library overlap. Attempt to insert a User Element in an occupied function table entry.

9.3 System Halt Codes

If the procedure for creating User Elements is not followed correctly, the controller will be stalled intentionally at the system start-up. This in order to avoid corruption of the application programs. First action then is to initiate the controller with the INIT. button and Auto mode selected. The controller will then restart in P2 mode.

Then

If any of the stall codes in sequence HALT60 to HALT64 occurs, pre-process your troublesome library program (or all) again following the installation procedure properly. The meaning of these codes are as follows.

60: One SDES AAB_FILE deleted in data base.

61: One CLIB AAB_FILE deleted in data base.

62: One SDES AAB_FILE not installed properly.

63: One CLIB AAB_FILE not installed properly.

If any of the stall codes in sequence HALT65 to HALT69 occurs, delete your User Element affected application programs, and pre-process your libraries again. Follow the installation procedure properly.

Appendix A System Messages

A.1 General

This appendix contains the specific system messages created by the User Element software module. Errors from this module are identified and displayed as messages of type 70 and task name starting with the letters **FAF**.

The system messages listed below are grouped in four tables related to the tasks they are likely to appear. The messages are sorted on trace code numbers in increasing order.

Messages related to internal system errors are clearly indicated in the column "Comments".

A.2 Command Monitor and Utilities (FAFA100)

Table A-1. Command Monitor and Utilities (MType 70)

Trace No (Hex)	Program/ Subprogram	Procedure/ Function	Parameter Description c=code, 1data1, 2data2	Comments
03051002	FAFA100	SdesCheckInst DBsource	c: fx\$ill_action 1: LF,LR - AAB_FILE 2: User Element No. in library.	Already defined User Element.
03051004	FAFA100	SdesCheckInst DBSource	c: fx\$in_param 1: error code 2: User Element No. in library.	Function table area is full. No place for more User Elements.
03051006	FAFA100	SdesCheckInst DBsource	c: fx\$ill_action 1: 0 2: 0	Incompatible system sw. User Elements created in a controller with diverse system sw.
03051008	FAFA100	CheckModId	c: fx\$ill_action 1: 0 2: 0	Incompatible system sw. User Elements created in a controller with diverse system sw.
0305100A	FAFA100	ClibCheckInst DBsource	c: fx\$ill_action 1: LF,LR - AAB_FILE 2: User Element No. in library.	CLIB's date older than SDES' date. PED library must be installed before the related library source code.
0305100C	FAFA100	ClibCheckInst DBsource	c: fx\$ill_action 1: LF,LR - AAB_FILE 2: User Element No. in library.	User Element size error.

Table A-1. Command Monitor and Utilities (MType 70) (Continued)

Trace No (Hex)	Program/ Subprogram	Procedure/ Function	Parameter Description c=code, 1data1, 2data2	Comments
0305100E	FAFA100	ClibCheckInst DBsource	c: fx\$ill_action 1: LF,LR - AAB_FILE 2: User Element No. in library.	Pointer to SDES lost.
03051012	FAFA100	ClibCheckInst DBsource	c: fx\$ill_action 1: LF,LR - AAB_FILE 2: User Element No. in library.	User Element name already in use.
03051014	FAFA100	ClibCheckInst DBsource	c: fx\$ill_action 1: 0 2: 0	Incompatible system sw. User Elements created in a controller with diverse system sw.
03051016	FAFA100	InstDBbuffer	c: fx\$ill_action 1: LF,LR - AAB_FILE 2: Buffer type	The controller must be in configuration mode during the installation.
03051018	FAFA100	InstDBbuffer	c: fx\$ill_action 1: LF,LR - AAB_FILE 2: Buffer type	The controller must be in configuration mode during the installation.
0305101A	FAFA100	InstDBbuffer	c: fx\$in_param 1: LF,LR - AAB_FILE 2: Buffer type	Illegal PED library.
0305101C	FAFA100	ForceDBinstall	c: fx\$in_param 1: LF,LR - AAB_FILE 2: 0	Library not complete. Force installation failed.
0305101E	FAFA100	ForceDBinstall	c: fx\$in_param 1: LF,LR - AAB_FILE 2: 0	Incorrect logical reference to AAB_DATA element(s).
03051022	FAFA100	ForceDBinstall	c: fx\$in_param 1: LF,LR - AAB_FILE 2: 0	No logical reference to AAB_DATA element(s).
03051024	FAFA100	ScanDBbuffer	c: fx\$in_param 1: LF,LR - AAB_FILE 2: Buffer type	Time-out, missing library end.
03051026	FAFA100	TreatDel AABfile	c: fx\$ill_action 1: 0 2: 0	One SDES has been deleted from the DB.

Table A-1. Command Monitor and Utilities (MType 70) (Continued)

Trace No (Hex)	Program/ Subprogram	Procedure/ Function	Parameter Description c=code, 1data1, 2data2	Comments
03051028	FAFA100	TreatDel AABfile	c: fx\$ill_action 1: 0 2: 0	One CLIB has been deleted from the DB.
0305102A	FAFA100	CommandScan AndTreat	c: fx\$ill_action 1: error code 2: command No.	The requested command No. was not performed. The related task is dormant.
0305102C	FAFA100	CommandScan AndTreat	c: fx\$ill_action 1: error code 2: SDES source identifier.	SDES identifier is out of range (0-99).
0305102E	FAFA100	CommandScan AndTreat	c: fx\$ill_action 1: error code 2: command No.	The requested command No. was not performed. The related task is dormant.
03051032	FAFA100	CommandScan AndTreat	c: fx\$ill_action 1: error code 2: PCPGM No.	Library program No. is out of range.
03051034	FAFA100	CommandScan AndTreat	c: fx\$ill_action 1: error code 2: Add. size percent.	The assigned spare factor value is out of range (0-100%).
03051035	FAFA100	CommandScan AndTreat	c: fx\$in_param 1: Logical record of file 2: Kilobyte to expand.	File expansion of AAB_FILE failed.
03051036	FAFA100	CommandScan AndTreat	c: fx\$ill_action 1: error code 2: command No.	Illegal command No. entered on FPS_WORK's property: version.
03051038	FAFA100	CommandScan AndTreat	c: fx\$ill_action 1: error code 2: 0	The controller must be in configuration mode.
03051039	FAFA100	AABDataMonitor	c: fx\$ill_action 1: error code 2: 0	The controller must be in configuration mode.
0305103A	FAFA100	CopyFnctab	c: fx\$ill_action 1: 0 2: 0	Internal system error. Function table copy error. ABORT.

Table A-1. Command Monitor and Utilities (MType 70) (Continued)

Trace No (Hex)	Program/ Subprogram	Procedure/ Function	Parameter Description c=code, 1data1, 2data2	Comments
0305103C	FAFA100	CopyFnctab	c: fx\$data_lost 1: 0 2: 0	Interface element creation error. ABORT.
0305103E	FAFA100	Initialize	c: fx\$in_param 1: 0 2: 0	Internal system error. Pointer to task parameter area not found. ABORT.
03051042	FAFA100	Main	c: fx\$in_param 1: 0 2: 0	Internal system error. DB buffer error. ABORT.
03051044	FAFA100	Main	c: 0 1: 0 2: 0	FPS_WORK, AAB_FILE created.
03051046	FAFA100	SdesInst DBsource	c: 0 1: 0 2: 0	SDES installed, AAB_FILE created.
03051048	FAFA100	ClibInst DBsource	c: 0 1: 0 2: 0	CLIB installed, AAB_FILE created.
03051050	FAFA100	SdesCheck-Inst DBsource	c: fx\$in_param 1: error code 2: User Element No. in library.	Attempt to insert a new User Element on occupied location in the function table.

A.3 Statement Description Pre-processor (FAFA200)

Table A-2. Statement Description Pre-processor (MType 70)

Trace No (Hex)	Program/ Subprogram	Procedure/ Function	Parameter Description c=code, 1data1, 2data2	Comments
03052000	FAFA200	Main	c: 0 1: start time (ms) 2: start day (day No.).	Start of pre-processing.
03052000	FAFA200	Main	c: 0 1: ending time (ms) 2: time used (ms).	End of pre-processing.
03052002	FAFA200	GetSymbol	c: fx\$in_param 1: error code 2: line No. in element's descr.	Missing terminal name.
03052004	FAFA200	GetSymbol	c: fx\$in_param 1: error code 2: line No. in element's descr.	Terminal name too long.
03052006	FAFA200	GetSymbol	c: fx\$in_param 1: error code 2: line No. in element's descr.	Missing quotation mark or illegal end of file.
03052008	FAFA200	GetSymbol	c: fx\$in_param 1: error code 2: line No. in element's descr.	Line end before comment end.
0305200A	FAFA200	GetSymbol	c: fx\$in_param 1: error code 2: line No. in element's descr.	Terminal name too long.
0305200C	FAFA200	GetSymbol	c: fx\$in_param 1: error code 2: line No. in element's descr.	Terminal name too long.
0305200E	FAFA200	GetSymbol	c: fx\$in_param 1: error code 2: line No. in element's descr.	Line end before comment end.
03052010	FAFA200	GetSymbol	c: fx\$in_param 1: error code 2: line No. in element's descr.	Word too long. (max. 255 characters)
03052012	FAFA200	ElementName	c: fx\$in_param 1: error code 2: line No. in element's descr.	Missing element name.

Table A-2. Statement Description Pre-processor (MType 70) (Continued)

Trace No (Hex)	Program/ Subprogram	Procedure/ Function	Parameter Description c=code, 1data1, 2data2	Comments
03052014	FAFA200	ElementName	c: fx\$in-param 1: error code 2: line No. in element's descr.	Element name too long.
03052016	FAFA200	ElementName	c: fx\$in_param 1: error code 2: line No. in element's descr.	Unexpected end of file.
03052018	FAFA200	ElementName	c: fx\$in_param 1: error code 2: line No. in element's descr.	Line end before comment end.
0305201A	FAFA200	ElementName	c: fx\$in_param 1: error code 2: line No. in element's descr.	Unexpected end of file.
0305201C	FAFA200	ElementName	c: fx\$in_param 1: error code 2: line No. in element's descr.	Word too long. (max. 255 characters)
0305201E	FAFA200	ElementName	c: fx\$in_param 1: error code 2: line No. in element	Element name too long.
03052020	FAFA200	ElementName	c: fx\$in_param 1: error code 2: line No. in element's descr.	Element name too long.
03052022	FAFA200	GetItem	c: fx\$in_param 1: error code 2: line No. in element's descr.	Word "terminal" not found.
03052024	FAFA200	GetItem	c: fx\$in_param 1: error code 2: line No. in element's descr.	Illegal entry. Word "function parameter" found.
03052026	FAFA200	GetItem	c: fx\$in_param 1: error code 2: line No. in element's descr.	Missing right parenthesis: ")".
03052028	FAFA200	GetItem	c: fx\$in_param 1: error code 2: line No. in element's descr.	Terminal number out of range.

Table A-2. Statement Description Pre-processor (MType 70) (Continued)

Trace No (Hex)	Program/ Subprogram	Procedure/ Function	Parameter Description c=code, 1data1, 2data2	Comments
0305202A	FAFA200	GetItem	c: fx\$in_param 1: error code 2: line No. in element's descr.	Number of members in Group item out of range.
0305202C	FAFA200	GetItem	c: fx\$in_param 1: error code 2: line No. in element's descr.	Illegal data type for members in Group item.
0305202E	FAFA200	GetItem	c: fx\$in_param 1: error code 2: line No. in element's descr.	Data type for members in Group item is missing.
03052030	FAFA200	GetItem	c: fx\$in_param 1: error code 2: line No. in element's descr.	Number of members in Group item is missing.
03052032	FAFA200	GetItem	c: fx\$in_param 1: error code 2: symbol	Array length out of range.
03052034	FAFA200	GetItem	c: fx\$in_param 1: error code 2: line No. in element's descr.	Missing length of array.
03052036	FAFA200	GetItem	c: fx\$in_param 1: error code 2: line No. in element's descr.	Illegal array syntax.
03052038	FAFA200	GetItem	c: fx\$in_param 1: error code 2: line No. in element	Unknown/missing data type.
0305203A	FAFA200	GetLine	c: fx\$in_param 1: error code 2: line No. in element's descr.	Wrong end of end line.
0305203C	FAFA200	GetLine	c: fx\$in_param 1: error code 2: line No. in element's descr.	Illegal entry. Call parameters in element head is not allowed.
0305203E	FAFA200	GetLine	c: fx\$in_param 1: error code 2: line No. in element's descr.	Illegal item(s) in start-line.

Table A-2. Statement Description Pre-processor (MType 70) (Continued)

Trace No (Hex)	Program/ Subprogram	Procedure/ Function	Parameter Description c=code, 1data1, 2data2	Comments
03052040	FAFA200	GetLine	c: fx\$in_param 1: error code 2: line No. in element's descr.	Several terminals in one line not allowed.
03052042	FAFA200	GetLine	c: fx\$in_param 1: error code 2: line No. in element's descr.	Terminal type expected.
03052044	FAFA200	GetLine	c: fx\$in_param 1: error code 2: line No. in element's descr.	Illegal item in User Element's description.
03052046	FAFA200	GetLine	c: fx\$in_param 1: error code 2: line nN	Illegal line-start.
03052048	FAFA200	GetLine	c: fx\$in_param 1: error code 2: line No. in element's descr.	Missing semicolon at line end.
0305204A	FAFA200	CreateGroup Descriptions	c: fx\$in_param 1: error code 2: term_number.	Too many terminals.
0305204C	FAFA200	EndGroup Descriptions	c: fx\$in_param 1: error code 2: 0	No terminals in User Element.
0305204E	FAFA200	EndGroup Descriptions	c: fx\$in_param 1: error code 2: line No. in element's descr.	Illegal line(s) between terminal lines.
03052050	FAFA200	CreateSDs	c: fx\$in_param 1: error code 2: 0	Empty PE file.
03052052	FAFA200	CreateSDs	c: fx\$in_param 1: error code 2: 0	End-line with no context.
03052054	FAFA200	CreateSDs	c: fx\$in_param 1: error code 2: line No. in element's descr.	Different User Element name in pc_element and end line.

Table A-2. Statement Description Pre-processor (MType 70) (Continued)

Trace No (Hex)	Program/ Subprogram	Procedure/ Function	Parameter Description c=code, 1data1, 2data2	Comments
03052056	FAFA200	CreateSDs	c: fx\$in_param 1: error code 2: line No. in element's descr.	Illegal "function parameter" line detected.
03052058	FAFA200	CreateSDs	c: fx\$in_param 1: error code 2: 0	End line is missing.
0305205A	FAFA200	CreateSDs	c: fx\$in_param 1: error code 2: 0	pc_element line is missing.
0305205C	FAFA200	CreatesSDs	c: fx\$in_param 1: error code 2: term_number	Terminal number not unique.
0305205E	FAFA200	CreatesSDs	c: fx\$in_param 1: error code 2: term_number	Terminal name not unique.
03052060	FAFA200	CreateSDs	c: fx\$in_param 1: error code 2: term_number	Terminal name not unique.
03052062	FAFA200	CreateSDs	c: fx\$in_param 1: error code 2: line No. in element's descr.	Unexpected end of file.
03052064	FAFA200	CreateSDs	c: fx\$in_param 1: error code 2: 0	No useful lines in PE file found.
03052066	FAFA200	CreateSDs	c: fx\$in_param 1: error code 2: err_count	Total number of errors detected during the last pre-processing.
03052068	FAFA200	SeaDouble Name	c: fx\$in_param 1: error code 2: User Element No. in library.	Already defined User Element in this library.
0305206A	FAFA200	SeaStandard Name	c: fx\$in_param 1: error code 2: User Element No. in library.	User Element name used by standard PC element.

Table A-2. Statement Description Pre-processor (MType 70) (Continued)

Trace No (Hex)	Program/ Subprogram	Procedure/ Function	Parameter Description c=code, 1data1, 2data2	Comments
0305206C	FAFA200	SeaSubroutine name	c: fx\$in_param 1: error code 2: User Element No. in library.	Already defined User Element in this library.
0305206E	FAFA200	ExCheck Install	c: fx\$in_param 1: error code 2: User Element No. in library.	Already defined interface element in this library.
03052070	FAFA200	ExCheck Install	c: fx\$in_param 1: error code 2: User Element No. in library.	Illegal changes of an existing User Element detected.
03052072	FAFA20	ExCheck Install	c: fx\$in_param 1: error code 2: User Element No. in library.	Internal system error. Function table is full. Attempt to install new elements is rejected.
03052074	FAFA200	NewCheck Install	c: fx\$in_param 1: error code 2: User Element No. in library.	Internal system error. Function table is full. Attempt to install new elements is rejected.
03052076	FAFA200	DBbuffer Exists	c: fx\$in_param 1: error code 2: ACB DB code	AAB_DATA expansion error.
03052078	FAFA200	Create DBbuffer	c: fx\$ill_action 1: error code 2: lf,lr to aab_data	FPS_WORK is corrupted. Delete and create a new FPS_WORK.
0305207A	FAFA200	Create DBbuffer	c: fx\$in_param 1: error code 2: ACB DB code	Creating a new AAB_FILE failed.
0305207C	FAFA200	Create DBbuffer	c: acb.ccode 1: error code 2: system error, ORD(cxyser)	Internal system error. Creating a new AAB_FILE failed.
0305207E	FAFA200	Initialize	c: fx\$task_param 1: 0 2: 0	Internal system error. Pointer to task parameter not found. ABORT.
03052080	FAFA200	Initialize	c: fx\$task_param 1: 0 2: 0	Internal system error. Pointer to common parameter area for User Element task not found.

Table A-2. Statement Description Pre-processor (MType 70) (Continued)

Trace No (Hex)	Program/ Subprogram	Procedure/ Function	Parameter Description c=code, 1data1, 2data2	Comments
03052082	FAFA200	Main	c: fx\$in_param 1: error code 2: 0	Memory allocation error.
03052084	FAFA200	Main	c: fx\$in_param 1: command No. 2: 0	Illegal command No. entered on FPS_WORK's property: version.
03052086	FAFA200	GetLine	c: fx\$in_param 1: error code 2: line No. in element's descr.	Text not allowed between terminal No. and name.
03052088	FAFA200	GetLine	c: fx\$in_param 1: error code 2: line No. in element's descr.	Wrong end of terminal line.
0305208A	FAFA200	GetLine	c: fx\$in_param 1: error code 2: line No. in element's descr.	Illegal item before word "type" detected.
0305208C	FAFA200	GetSymbol	c: fx\$in_param 1: error code 2: line No. in element's descr.	Line ended within illegal name.
0305208E	FAFA200	GetSymbol	c: fx\$in_param 1: error code 2: line No. in element's descr.	Unexpected end of line.
03052090	FAFA200	GetSymbol	c: fx\$in_param 1: error code 2: line No. in element's descr.	Unexpected end of line.

A.4 Code Library Pre-processor (FAFA300)

Table A-3. Code Library Pre-processor (MType 70)

Trace No (Hex)	Program/ Subprogram	Procedure/ Function	Parameter Description c=code, 1data1, 2data2	Comments
03053005	FAFA300	ReadOCElement	c: fx\$in_param 1: pointer to offset in object code table 2: pointer to descr. of get put chain in object code table	Illegal connection to control module executed by an other interpreter.
03053006	FAFA300	TrimAnd copyT.	c: fx\$in_param 1: error code 2: error in dayy02	Terminal data type error. Retranslate the CONTRM affected.
03053008	FAFA300	TreatTotalC	c: fx\$in_param 1: error code 2: PC element type	Library program error.
0305300A	FAFA300	TreatTotalC	c: fx\$in_param 1: error code 2: PC element type	Library program main structure error.
0305300C	FAFA300	TreatTotalC	c: fx\$in_param 1: error code 2: CONTRM No.	Library program structure error.
0305300E	FAFA300	SeaDouble Ind.	c: fx\$in_param 1: error code 2: User Element No. in library.	Already defined User Element in this library.
03053010	FAFA300	ContCheck Ins.	c: fx\$ill_action 1: error code 2: entry in function table for User Element	User Element requires to much size. Function table size exceeded.
03053012	FAFA300	ContCheck Ins-	c: fx\$ill_action 1: error code 2: entry in function table for User Element	Pointer to SDES is lost. Execute force install.
03053014	FAFA300	DBbuffer Exis.	c: fx\$in_param 1: error code 2: ACB DB code	AAB_DATA expansion error.
03053016	FAFA300	CreateDB buffer	c: fx\$in_param 1: error code 2: ACB DB code	Creating a new AAB_FILE failed.

Table A-3. Code Library Pre-processor (MType 70) (Continued)

Trace No (Hex)	Program/ Subprogram	Procedure/ Function	Parameter Description c=code, 1data1, 2data2	Comments
03053018	FAFA300	CreateDB buffer	c: fx\$in_param 1: error code 2: system error, ORD(cxsyserr)	Creating a new AAB_FILE failed.
0305301A	FAFA300	Initialize	c: fx\$task_param 1: 0 2: 0	Internal system error. Pointer to task parameter area not found.
0305301C	FAFA300	Initialize	c: 1: 2:	Internal system error. Abort task.
0305301E	FAFA300	Initialize	c: fx\$task_param 1: 0 2: 0	Internal system error. Pointer to common parameter area not found.
03053020	FAFA300	Initialize	c: 1: 2:	Internal system error. Abort task.
03053022	FAFA300	Main	c: fx\$NoError 1: start time (ms) 2: start day (day No.)	Start of pre-processing.
03053024	FAFA300	Main	c: fx\$in_param 1: error code 2: PC element type	Internal system error. Library program object code table corrupt.
03053025	FAFA300	Main	c: fx\$in_param 1: error code 2: PC element type	Library program structure error.
03053026	FAFA300	Main	c: fx\$in_param 1: error code 2: local data tab. size	Memory allocation error. Local data area full.
03053028	FAFA300	Main	c: fx\$in_param 1: error code 2: PC table status.	Internal system error. Pointers to PC system tables lost.
0305302A	FAFA300	Main	c: fx\$in_param 1: error code 2:	Memory allocation error. Re- dimension DB area.

Table A-3. Code Library Pre-processor (MType 70) (Continued)

Trace No (Hex)	Program/ Subprogram	Procedure/ Function	Parameter Description c=code, 1data1, 2data2	Comments
0305302C	FAFA300	Main	c: fx\$NoError 1: ending time (ms) 2: time used (ms).	End of pre-processing.
0305302E	FAFA300	Main	c: fx\$in_param 1: error code 2: command No.	Illegal start command entered on FPS_WORK's property: version.

A.5 Common Pascal Routines (FAFB01)

Table A-4. Common Pascal Routines (MType 70)

Trace No	Program/ Subprogram	Procedure/ Function	Parameter	Comments
03050102	FAFB01	ExpandAABFile	c: fx\$in_param 1: If,lr failed element 2:	Failed to add an AAB_FILE. Memory occupied or not enough memory.
03050108	FAFB01	ExpandAABFile	c: fx\$in_param 1: last If,lr accessed when fail 2:	Internal error when trying to allocate AAB_FILES.

Appendix B User Element Related Files in FCB Directory Structure

B.1 General

Directories and files, relevant when working on creation of User Elements are shown in the figure below. Also files created during the FCB installation are shown.

Directories subordinate to user defined projects and nodes are automatically created each time a new project or node is defined in the Application Builder. These directories are indicated in capital letters.

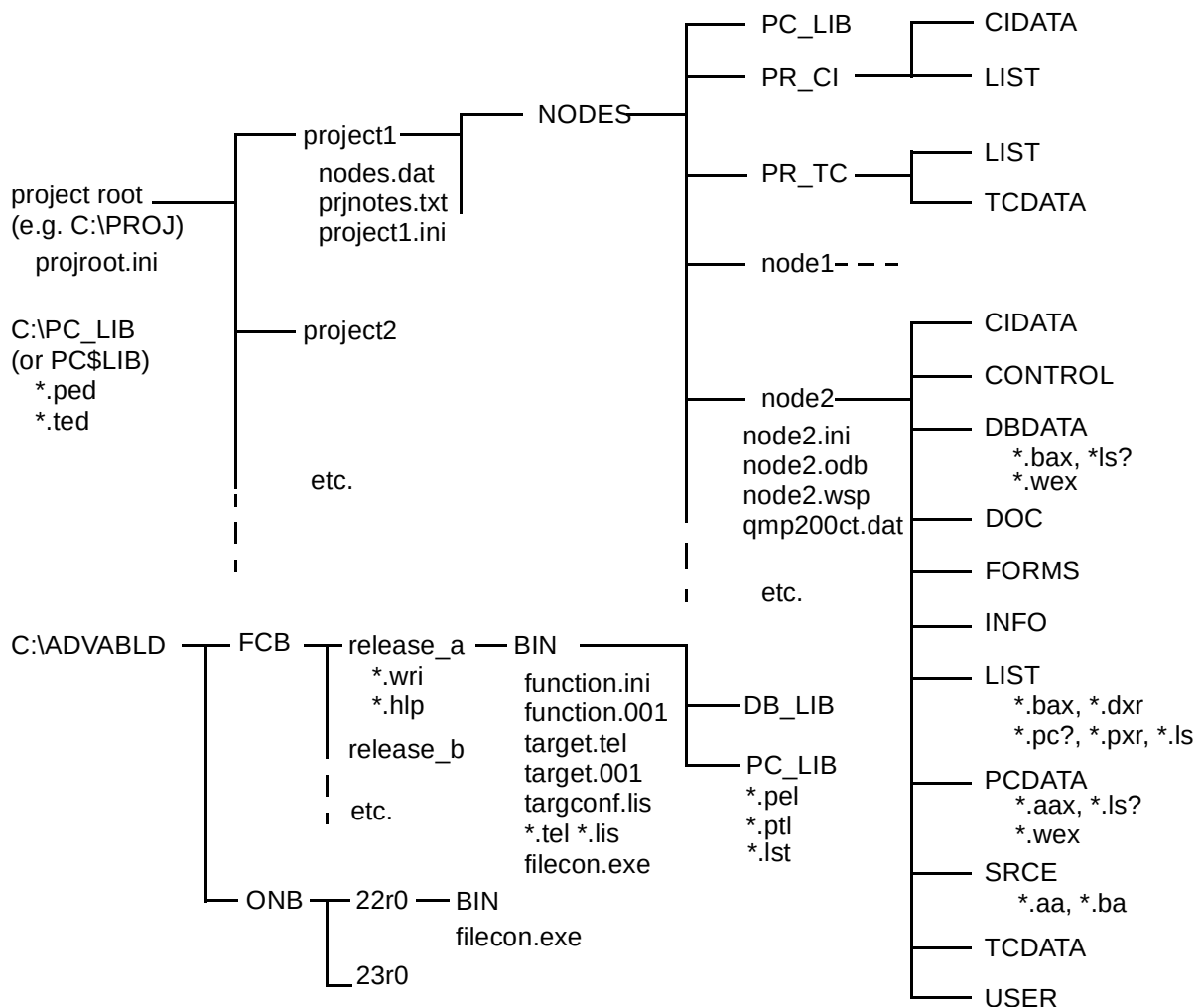


Figure B-1. FCB Directory Structure with Files Relevant when Creating User Elements

Short Description of the Indicated Files.

- Files in directory for PED libraries **C:\PC_LIB** (or utilized **PC\$LIB** variable)
 - *.PED = PC element description libraries
 - *.TED = Target description definitions
 - Files in release directories **C:\ADVABLD\FCB\release_*** :
 - *.wri = FCB release notes
 - *.hlp = FCB related “read me” files
 - Files in release binary directories **C:\ADVABLD\FCB\release_*.BIN** :
 - function.ini = initialization file with supported node types, options and libraries
 - function.001 = backup of function.ini. Copied manually by the user immediate after FCB installation to preserve information on standard node types and libraries.
 - filecon.exe = command for conversion of PED libraries. Resulting file: PED library for interface elements.
 - target.tel = target description file in binary format. Contains description of all supported targets. Modified by Setup Add-Ons after each successful installation of TED definitions.
 - target.001 = backup of previous target.tel created automatically by Setup Add-Ons.
 - targconf.lis = log file of Setup Add-Ons
 - *.tel = target description files in binary format. Contains description of a target modified by the user. Created by Setup Add-Ons after a successful installation of TED definitions and PED libraries for the target concerned.
 - *.lis = list file of Setup Add-Ons. List of detected errors in a TED file during the compilation.
- ...\PC_LIB :**
- *.pel = PED libraries in binary format.
 - *.ptl = binary data on execution time and memory consumption for elements in PED libraries.
 - *.lst = list file of Setup Add-Ons. List of detected errors in a PED file during the compilation.
- Files in release binary directories **C:\ADVABLD\ONB\22r0\BIN and ... \23r0\BIN:**
 - filecon.exe = command for conversion of PED libraries. Resulting file: PED library for interface elements.
- Files in project directories **project_root** :
 - *.ini = information file for <project_root>

- Files in project directories **project_root\project*** :
 - nodes.dat = file listing the nodes of <project_root\project>
 - prjnotes.txt = file with creation note for <project*>
 - <project>.ini = defaults for <project>
 - Files in node directories **project_root\project*\NODES\node*** :
 - *.ini = information file for <node>
 - *.odb = FCB's internal data representation of <node> (resolved and expanded)
 - *.wsp = general work space parameters saved at the end of every FCB session
 - qmp200ct.dat = file containing the configurable scan times
- ...\DBDATA :**
- *.bax = AMPL source code files for the DB. Created when test of control solutions is required
 - *.ls? = list files generated by the DB backtranslator
 - *.wex = files containing detected error locations in the source *.bax files
- ...\LIST :**
- *.db = DB listing generated by printing to file *
 - *.dxr = DB cross-reference listing generated by printing to file *
 - *.pct = tree diagram (document) generated by printing to file *
 - *.pcf = function chart (document) generated by printing to file *
 - *.pxr = PC cross-reference listing generated by printing to file *
 - *.syl = symbol listing generated by printing to file*
- ...\PCDATA:**
- *.aax = AMPL source code files for PC programs (application and library programs)
 - *.ls? = list files generated by the PC backtranslator
 - *.wex = files containing detected error locations in the source files *.aax
- ...\SRCE :**
- *.aa = AMPL source code files for PC programs (in compressed form)
 - *.ba = AMPL source code files for the DB (in compressed form)

Appendix C Group of Characters Applied in PED and TED Files

C.1 General

The group of characters allowed for use in PED and TED files is based on the ASCII character set within range 32 - 126.

In order to simplify the explanation we will classify all the actual characters in following groups:

LETTER	Uppercase US/English letters A ... Z
Letter	Lowercase US/English letters a ... z
Digit	Digits 0 ... 9
Separator	_ - \
Wildcard	* # \$?
Terminal	< > [] _ + - / =
Special	. ! : { } () / \ + - ^ ' ~ % & @ SPACE
Reserved	, " ; '

Where these groups of characters are used is illustrated in the [Table C-1](#).

Table C-1. Allowed Characters in PED and TED Definition Files

PED File	TED File	Allowed Character Groups
Not applicable	"option name" in quotations	LETTER + Letter + Digit + Wildcard + Special
	Option name	LETTER + Letter + Digit + Wildcard + Special except for SPACE
	Option version	
LIBNAME	pc_element_library name	The first character must not be - (hyphen)
LIBVERSION	pc_element_library version	
Not applicable	Comment	LETTER + Letter + Digit + Wildcard + Special
Comment enclosed in (* *)	Not applicable	All groups allowed.
User Element name	Not applicable	LETTER + Digit + Separator + Wildcard The first character must be LETTER
Terminal name	Not applicable	LETTER + Digit + Separator + Terminal



3BSE 009 739R0001 Rev A

February 1998